TMS320C1x User's Guide

2563968-9721 revision *
July 1991

TEXAS INSTRUMENTS

important notice

Texas Instruments Incorporated (TI) reserves the right to make changes to its products or to discontinue any semiconductor product or service without notice, and advises its customers to obtain the latest version of relevant information to verify, before placing orders, that the information being relied on is current.

TI warrants performance of its semiconductor products and related software to current specifications in accordance with TI's standard warranty. Testing and other quality control techniques are utilized to the extent TI deems necessary to support this warranty. Specific testing of all parameters of each device is not necessarily performed, except those mandated by government requirements.

Please be aware that TI products are not intended for use in life-support appliances, devices, or systems. Use of TI product in such applications requires the written approval of the appropriate TI officer. Certain applications using semiconductor devices may involve potential risks of personal injury, property damage, or loss of life. In order to minimize these risks, adequate design and operating safeguards should be provided by the customer to minimize inherent or procedural hazards. Inclusion of TI products in such applications is understood to be fully at the risk of the customer using TI devices or systems.

TI assumes no liability for applications assistance, customer product design, software performance, or infringement of patents or services described herein. Nor does TI warrant or represent that any license, either express or implied, is granted under any patent right, copyright, mask work right, or other intellectual property right of TI covering or relating to any combination, machine, or process in which such semiconductor products or services might be or are used.

Read This First

How to Use This User's Guide

The purpose of this user's guide is to serve as a reference book for the first-generation TMS320 digital signal processors. Chapters 2 through 4 provide specific information on architecture and operation of these devices. Appendix A describes the electrical specifications and mechanical data information for the commercial devices of the TMS320C1x generation. Appendix B describes the same data for the military devices of the SMJ320C1x generation. Appendices C through F provide information on mask options, quality and reliability, device support, and DSP-related devices.

Note:

Throughout this book, 'C14 designates all devices with a 14 suffix (C14/E14/P14), 'C15 all devices with a 15 suffix (C15/E15/LC15/P15), and 'C17 all devices with a 17 suffix (C17/E17/LC17/P17), unless otherwise noted.

This book contains the following chapters:

Chapter 1 **Introduction**
Describes the TMS320 family product line evolution and their functional and perfomance characteristics.

Chapter 2 **Pinouts and Signal Descriptions**
Drawings of the packages for TMS320C1x devices. Functional listings of the signals, their pin locations, and descriptions.

Chapter 3 **Architecture**
TMS320C1x design description, hardware components, and device operation. Functional block diagrams and internal hardware summary table.

Chapter 4 **Assembly Language Instructions**
Addressing modes and format descriptions. Instruction set summary listed according to function. Alphabetized individual instruction descriptions with examples.

Chapter 5 **Software Applications**
Software application examples for the use of variousTMS320C1x instruction set features.

Chapter 6 **Hardware Applications**
Hardware design techniques and application examples for interfacing to codecs, external memory, or common 4-/8-/16-/32-bit microcomputers and microprocessors.

Appendix A **TMS320C1x Digital Signal Processors**
Electrical specifications,timing, and mechanical data for all TMS320C1x devices.

Appendix B **SMJ320C1x Digital Signal Processors**
Electrical specifications,timing, and mechanical data for all SMJ320C1x devices.

Appendix C **ROM Codes**
Discussion of ROM codes (mask options) and the procedure for implementation.

Appendix D **Quality and Reliability**
Discussion of Texas Instruments quality and reliability criteria for evaluating performance.

Appendix E **Development Support**
Information about the hardware and software available to support the TMS320C1x devices.

Appendix F **Analog Interface Peripherals and Applications**
Digital /analog converters and interface devices for DSP applications in multimedia, telecommunications, speech synthesis, servo control, modems, and consumer electronics.

Appendix G **Programming the EPROM Cell**
Programming, verifying, reading, and protecting of the TMS320E1x/P1x EPROM cells.

Related Documentation

General Digital Signal Processing:

Antoniou, Andreas, *Digital Filters: Analysis and Design.* New York, NY: McGraw-Hill Company, Inc., 1979.

Brigham, E. Oran, *The Fast Fourier Transform.* Englewood Cliffs, NJ: Prentice-Hall, Inc., 1974.

Burrus, C.S., and Parks, T.W., *DFT/FFT and Convolution Algorithms.* New York, NY: John Wiley and Sons, Inc., 1984.

Digital Signal Processing Applications with the TMS320 Family, Englewood Cliffs, NJ: Texas Instruments, 1986; Prentice-Hall, Inc., 1987.

Gold, Bernard, and Rader, C. M., *Digital Processing of Signals.* New York, NY: McGraw-Hill Company, Inc., 1969.

Hamming, R.W., *Digital Filters.* Englewood Cliffs, NJ: Prentice-Hall, Inc., 1977.

IEEE ASSP DSP Committee (Editor), *Program for Digital Signal Processing.* New York, NY: IEEE Press, 1979.

Jackson, Leland B., *Digital Filters and Signal Processing.* Hingham, MA: Kluwer Academic Publishers, 1986.

Jones, D.L., and Parks, T.W., *A Digital Signal Processing Laboratory Using the TMS32010.* Englewood Cliffs, NJ: Prentice-Hall, Inc., 1987.

Lim, Jae, and Oppenheim, Alan V., *Advanced Topics in Signal Processing.* Englewood Cliffs, NJ: Prentice-Hall, Inc., 1988.

Morris, L. Robert, *Digital Signal Processing Software.* Ottawa, Canada: Carleton University, 1983.

Oppenheim, Alan V., (Editor), *Applications of Digital Signal Processing.* Englewood Cliffs, NJ: Prentice-Hall, Inc., 1978.

Oppenheim, Alan V. and Schafer, R.W., *Digital Signal Processing.* Englewood Cliffs, NJ: Prentice-Hall, Inc., 1975.

Oppenheim, Alan V., and Willsky, A.N., with Young, I.T., *Signals and Systems.* Englewood Cliffs, NJ: Prentice-Hall, Inc., 1983.

Parks, T.W., and Burrus, C.S., *Digital Filter Design.* New York, NY: John Wiley and Sons, Inc., 1987.

Rabiner, Lawrence R., and Gold, Bernard, *Theory and Application of Digital Signal Processing.* Englewood Cliffs, NJ: Prentice-Hall, Inc., 1975.

Treichler, J.R., Johnson, Jr., C.R., and Larimore, M.G., *Theory and Design of Adaptive Filters*. New York, NY: John Wiley and Sons, Inc., 1987.

Speech:

Gray, A.H., and Markel, J.D., *Linear Prediction of Speech*. New York, NY: Springer-Verlag, 1976.

Jayant, N.S., and Noll, Peter, *Digital Coding of Waveforms*. Englewood Cliffs, NJ: Prentice-Hall, Inc., 1984.

Papamichalis, Panos, *Practical Approaches to Speech Coding*. Englewood Cliffs, NJ: Prentice-Hall, Inc., 1987.

Rabiner, Lawrence R., and Schafer, R.W., *Digital Processing of Speech Signals*. Englewood Cliffs, NJ: Prentice-Hall, Inc., 1978.

Image Processing:

Andrews, H.C., and Hunt, B.R., *Digital Image Restoration*. Englewood Cliffs, NJ: Prentice-Hall, Inc., 1977.

Gonzales, Rafael C., and Wintz, Paul, *Digital Image Processing*. Reading, MA: Addison-Wesley Publishing Company, Inc., 1977.

Pratt, William K., *Digital Image Processing*. New York, NY: John Wiley and Sons, 1978.

Digital Control Theory:

Astrom, K., and Wittenmark, B., *Computer Controlled Systems*. Englewood Cliffs, NJ: Prentice-Hall, Inc., 1984.

Iserman, R., *Digital Control Systems*. New York, NY: Springer-Verlag, 1981.

Jacquot, R., *Modern Digital Control Systems*. New York, NY: Marcel Dekker, Inc., 1981.

Katz, P., *Digital Control Using Microprocessors*. Englewood Cliffs, NJ: Prentice-Hall, Inc., 1981.

Kuo, B.C., *Digital Control Systems*. New York, NY: Holt, Reinholt and Winston, Inc.,1980.

Moroney, P., *Issues in the Implementation of Digital Feedback Compensators*. Cambridge, MA: The MIT Press, 1983.

Phillips, C., and Nagle, H., *Digital Control System Analysis and Design*. Englewood Cliffs, NJ: Prentice-Hall, Inc., 1984.

Style and Symbol Conventions

This document uses the following conventions.

- Program listings and examples are shown in `Courier` font, which is similar to a typewriter's.

  Here is a sample code listing segment:

```
DSP COFF Assembler Version 5.10 Tue Apr 23 17:38:39 1991
 (c) Copyright 1987, 1989, Texas Instruments Incorporated
PID CONTROL                                        PAGE   1
0002                *
0003                * THIS ROUTINE IMPLEMENTS A PID ALGORITHM.
0004                *
0005   0000   UN       .set   0   ;OUTPUT OF CONTROLLER
0006   0001   E0       .set   1   ;LATEST ERROR SAMPLE
0007   0002   E1       .set   2   ;PREVIOUS ERROR SAMPPLE
0008   0000   UN       .set   0   ;OUTPUT OF CONTROLLER
0009   0001   E0       .set   1   ;LATEST ERROR SAMPLE
0010   0002   E1       .set   2   ;PREVIOUS ERROR SAMPLE
0011   0003   E2       .set   3   ;OLDEST ERROR SAMPLE
0012   0004   K1       .set   4   ;GAIN CONSTANT
0013   0005   K2       .set   5   ;GAIN CONSTANT
0014   0006   K3       .set   6   ;GAIN CONSTANT
0015
No Errors,  No Warnings
```

- In syntax descriptions, the instruction and associated parameters are shown in the same font as regular text. In the description, the function names and the parameters are shown in *italic* typeface. The information that you or the software provide is also shown in this face. For example, in:

  [*label*] ADD {*|*+|*–} [*shift* [,*next ARP*]]

- The square brackets ([and]) identify optional information that you provide; you don't enter the brackets themselves. In the example, instead of *label* you provide a name for the label.

- Braces ({ and }) indicate a list. The symbol | (read as *or*) separates items within the list. In the example:

  {*|*+|*–}

  is an operand with three choices: `*`, `*+`, or `*-`.

  As the list is not enclosed in square brackets, you must choose one item from the list.

  In the example, for the instruction ADD you must specify one parameter from the list {*|*+|*–}. The parameters [*shift*[,*next ARP*]] are optional; if you use them, you replace them with the actual value of the parameters. That is, if you use *shift*, you specify a number between 0 and 15. If you use *next ARP* you specify a 0 or a 1.

Customer Support

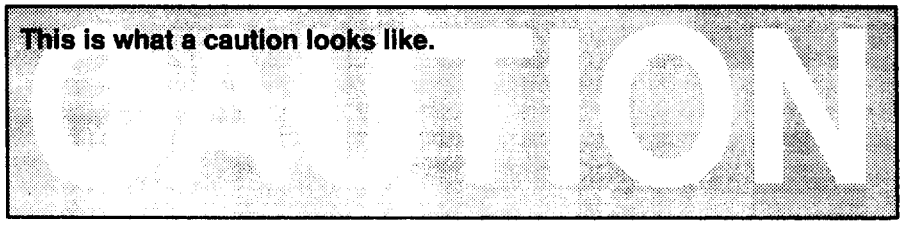
If you need assistance or wish to report problems with your TMS320 software call the TMS320 DSP hotline at (713) 274-2320 (8 AM–5 PM CST). To access the DSP bulletin board service dial (713) 274-2323 (24 hrs). Set your modem to 2400/1200,8,N,1 and follow the system prompts to sign in.

Information About Cautions

This book may contain cautions

- A **caution** describes a situation that could potentially damage your software or equipment.

This is what a caution looks like.



Trademarks

Aegis and *Apollo* are trademarks of Apollo Computer, Inc.
DEC, VAX, and VMS are trademarks of Digital Equipment Corp.
MS, and *MS-DOS* are trademarks of Microsoft Corp.
PC-DOS is a trademark of IBM Corp.
Sun 3 is a trademark of Sun Microsystems, Inc.
UNIX is a registered trademark of AT&T Bell Laboratories.
XDS is a trademark of Texas Instruments Incorporated.

Contents

Figures

Tables

Examples

Chapter 1

Introduction

The TMS320 family of 16/32-bit single-chip digital signal processors combines the flexibility of a high-speed controller with the numerical capability of an array processor, offering an inexpensive alternative to custom VLSI and multichip bit-slice processors.

The TMS32010, the first digital signal processor in the TMS320 family, was introduced in 1983. During that year, the TMS32010 was named Product of the Year by the magazine, *Electronic Products*. The TMS320 family now consists of five generations of processors: TMS320C1x, TMS320C2x, TMS320C3x, TMS320C4x, and TMS320C5x. Many features are common among these generations. Some specific features are added to each processor to provide different price/performance alternatives. Software compatibility is maintained throughout the family to protect your software investment. Each processor has software and hardware tools to facilitate design.

Note:

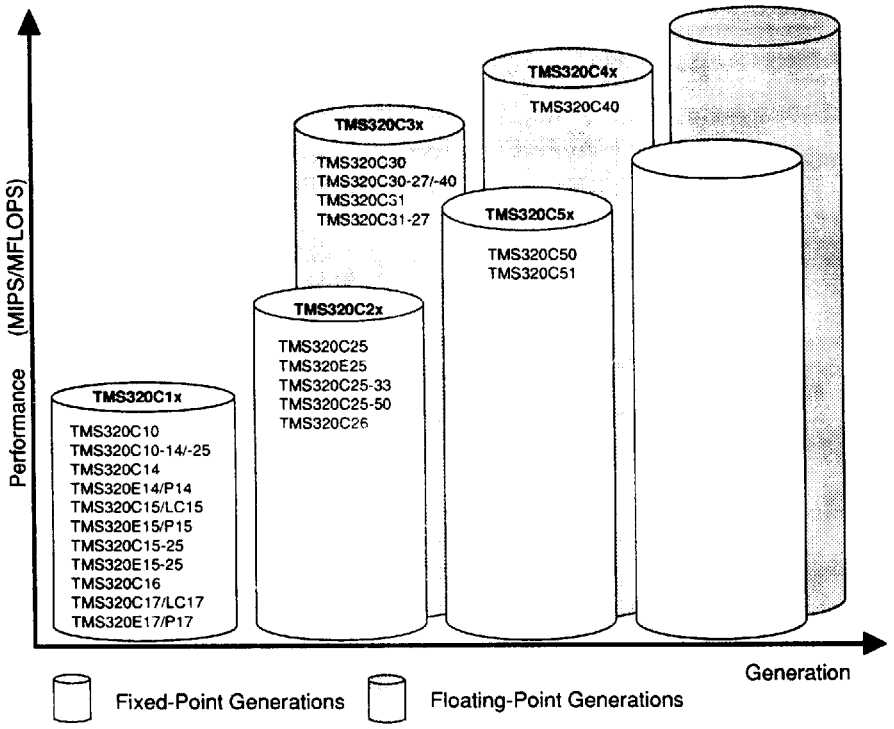
Throughout this book, 'C14 designates all devices with a 14 suffix (C14/E14/P14), 'C15 all devices with a 15 suffix (C15/E15/LC15/P15), and 'C17 all devices with a 17 suffix (C17/E17/LC17/P17), unless otherwise noted.

This chapter discusses the following

1.1 The TMS320 Family

The TMS320C1x generation combines the high performance and specialized features necessary in digital signal processing (DSP) applications. TI also offers an extensive program of development support, including hardware and software development tools, product documentation, textbooks, newsletters, DSP design workshops, and a variety of application reports.See Appendix E for a discussion of the wide range of development tools available.

Figure 1–1. TMS320 Device Evolution



Throughout this document, the devices within the TMS320C1x generation are collectively referred to as TMS320C1x. The specific members of the TMS320C1x generation include

- TMS320C10, a CMOS 20-MHz version of the TMS32010
- TMS320C10-14, a 14-MHz version of the TMS320C10
- TMS320C10-25, a 25-MHz version of the TMS320C10
- TMS320C14, a TMS320C15 designed for a control system
- TMS320E14, an EPROM version of the TMS320C14
- TMS320P14, a one-time programmable version of the TMS320E14
- TMS320C15, a TMS320C10 with expanded ROM and RAM
- TMS320C15-25, a 25-MHz version of the TMS320C15
- TMS320LC15, a low-power version of the TMS320C15
- TMS320E15, an EPROM version of the TMS320C15
- TMS320E15-25, a 25-MHz version of the TMS320E15

- TMS320P15, a one-time programmable version of the TMS320E15
- TMS320C16, a 35-MHz expanded memory version of the TMS320C15
- TMS320C17, a TMS320C15 with serial and coprocessor ports
- TMS320LC17, a low-power version of the TMS320C17
- TMS320E17, an EPROM version of the TMS320C17
- TMS320P17, a one-time programmable version of the TMS320E17

Figure 1–1 shows the different devices in each generation of the TMS320 family. Plans for expansion of the TMS320C1x generation include more spinoffs of the existing generations, as well as more powerful future generations of digital signal processors.

1.2 General Description

The combination of a Harvard-type architecture (separate program and data buses) and special digital signal processing (DSP) instruction set give the TMS320C1x generation speed and flexibility to execute up to 8.77 MIPS (million instructions per second). While other processors implement functions through software or microcode, the TMS320C1x generation optimizes performance by implementing functions within the hardware. This hardware-intensive approach supplies the design engineer with power previously unavailable on a single chip.

Table 1–1 presents an overview of the TMS320C1x generation of DSPs, including technology, memory, I/O, cycle timing, package type, and military support.

Table 1–1. TMS320C1x Overview

Device	Tech.	Memory				I/O		Cycle	Package (1)		
		RAM	ROM	EPROM	Prog.	Serial	Parallel	(ns)	DIP	PLCC	Quad
TMS320C10 (2)	CMOS	144	1.5K	–	4K	–	8x16	200	40	44	–
TMS320C10-14	CMOS	144	1.5K	–	4K	–	8x16	280	40	44	–
TMS320C10-25	CMOS	144	1.5K	–	4K	–	8x16	160	40	44	–
TMS320C14 (3)	CMOS	256	4K	–	4K	1	7x16 (4)	160	–	68	–
TMS320E14 (3)	CMOS	256	–	4K	4K	1	7x16 (4)	160	–	–	68 CER
TMS320P14	CMOS	256	–	4K	4K	1	7x16 (4)	160	–	68	–
TMS320C15 (3)	CMOS	256	4K	–	4K	–	8x16	200	40	44	44QFP
TMS320C15-25	CMOS	256	4K	–	4K	–	8x16	160	40	44	–
TMS320E15 (3)	CMOS	256	–	4K	4K	–	8x16	200	40	–	44 CER
TMS320E15-25	CMOS	256	–	4K	4K	–	8x16	160	40	–	44 CER
TMS320LC15	CMOS	256	4K	–	4K	–	8x16	200	40	44	–
TMS320P15	CMOS	256	–	4K	4K	–	8x16	200	40	44	–
TMS320C16	CMOS	256	8K	–	64K		8x16	114	–	–	64 QFP
TMS320C17	CMOS	256	4K	–	–	2	6x16 (5)	200	40	44	–
TMS320E17	CMOS	256	–	4K	–	2	6x16 (5)	200	40	–	44 CER
TMS320LC17	CMOS	256	4K	–	–	2	6x16 (5)	200	40	44	–
TMS320P17	CMOS	256	–	4K	–	2	6x16 (5)	200	40	44	–

Notes: 1) DIP = dual in-line package. PLCC = plastic-leaded chip carrier. CER-QUAD = ceramic-leaded chip carrier. QFP = plastic quad flat pack).
2) Military version available.
3) Military versions planned; contact nearest TI Field Sales Office for availability.
4) On-chip 16-bit I/O, four capture inputs, and six compare outputs are available.
5) On-chip 16-bit coprocessor interface is optional by pin selection.

All devices are processed in CMOS technology.

- The **TMS320C10** microprocessor is capable of achieving a 16 x 16-bit multiply in a single 200-ns cycle. On-chip data memory of 144 words is avail-

able. Up to 4K words of off-chip program memory can be executed at full speed. Because of its low-power dissipation (165 mW), the TMS320C10 is ideal for power-sensitive applications such as digital telephony and portable consumer products. A masked ROM option is available.

- The **TMS320C10-14**, a 14-MHz version of the TMS320C10, provides a low-cost alternative for DSP applications not requiring the maximum operating frequency of the TMS320C10. The device can execute 3.5 million instructions per second and has a 280-ns instruction cycle time.

- The **TMS320C10-25**, a 25-MHz version of the TMS320C10, has a 160-ns instruction cycle time. Its lower power and higher speed make it well suited for high-performance DSP applications.

- The **TMS320C14**, **TMS320E14**, and **TMS320P14** microcontrollers have an instruction cycle time of 160 ns, 256 words of on-chip RAM, and 4K words of on-chip program ROM (TMS320C14) or EPROM (TMS320E14). The TMS320C14 and the TMS320E14 feature an event manager with four capture inputs and six compare outputs, a bit-selectable I/O port, a serial port with programmable protocols and timer, a watchdog timer, and two general-purpose timers. These devices are object-code compatible with the TMS320C10. The TMS320P14 is a one-time programmable version of the TMS320E14. The TMS320C14 is designed for control system applications and comprises the first devices that combine the high performance of a DSP with the on-chip peripherals of a microcontroller.

- The **TMS320C15, TMS320E15**, **TMS320LC15**, and **TMS320P15** are fully object-code and pin-for-pin compatible with the TMS320C10. Each offers an expanded on-chip RAM of 256 words and an on-chip program ROM (TMS320C15) or EPROM (TMS320E15) of 4K words. the TMS320C15-25 and TMS320E15-25 are 160 ns versions. The TMS320LC15 is a low-power (3-volt) version of the TMS320C15; the TMS320P15 is a one-time programmable version of the TMS320E15.

- The **TMS320C16** microcontroller has 256 words of on-chip data RAM, 8K words of on-chip ROM, and 64K of external program space. It has an instruction time of 114 ns and is capable of executing 8.77 MIPs (million instructions per second)

- The **TMS320C17**, **TMS320E17**, **TMS320LC17**,and **TMS320P17**are dedicated microcomputers. Each offers 256 words of on-chip RAM and 4K words of on-chip program ROM (TMS320C17) or EPROM (TMS320E17 and TMS320P17). The TMS320C17 features a dual-channel serial interface, on-chip companding hardware (μ-law/A-law), a serial port timer, and a latched16-bit coprocessor port for direct microprocessor I/O interface. The devices are object-code compatible with the TMS320C10. The TMS320P17 is a one-time programmable version of the TMS320E17. The TMS320LC17 is a low power (3-volt) version of the TMS320C17.

1.3 Key Features

The key features of the TMS320C1x devices are listed below.

- Instruction cycle timing:
  - 114 ns
    - TMS320C16
  - 160 ns
    - TMS320C10-25
    - TMS320C14
    - TMS320C15-25/E15-25
  - 200 ns
    - TMS320C10
    - TMS320C15/LC15
    - TMS320C17/LC17
  - 280 ns
    - TMS320C10-14
- 144-/256-word on-chip data RAM
- 1.5K-/4K-word on-chip program ROM (TMS320C1x except 'C16)
- 8K-word on-chip program ROM (TMS320C16)
- 4K-word on-chip program EPROM (TMS320E/P1x)
- EPROM code protection for copyright security
- 64K-word total external memory at full speed
  - TMS320C16
- 4K-word total external memory at full speed
  - TMS320C10
  - TMS320C14
  - TMS320C15
- 16-bit bidirectional data bus at 50-Mbps transfer rate
- 32-bit ALU/accumulator
- 16 × 16-bit parallel multiplier with a 32-bit product
- 0- to 16-bit barrel shifter
- On-chip clock generator
- Eight input and eight output channels
- Dual-channel serial port with timer (TMS320C17)
- Direct interface to combo-codecs (TMS320C17)
- On-chip μ-law/A-law companding hardware (TMS320C17)
- 16-bit coprocessor interface (TMS320C17)
- 16-pin bit-selectable I/O ports (TMS320C14)
- Serial port with programmable protocols (TMS320C14)
- Event manager with capture inputs and compare outputs (TMS320C14)

- Four independent timers (TMS320C14)
  - General-purpose (2
  - Serial port
  - Watchdog
- Fifteen external/internal interrupts (TMS320C14)
- Single 5-V supply (3-V for TMS320LC15/LC17)
- Device packaging:
  - 40-pin DIP
    - TMS320C10
    - TMS320C15/E15/LC15/P15
    - TMS320C17/E17/LC17/P17
  - 44-lead PLCC
    - TMS320C10
    - TMS320C/LC/15/17
    - TMS320C/LC/15/17
  - 68-lead PLCC
    - TMS320C14/P14
  - 44-lead CER-QUAD
    - TMS320E15
    - TMS320E17
  - 68-lead CER-QUAD
    - TMS320E14
  - 64-lead quad flat pack
    - TMS320C16
- CMOS Technology
- Commercial and military versions available.

1.3.1 Typical Applications

The TMS320C1x generation's unique versatility and realtime performance facilitates flexible design approaches in a variety of applications. In addition, TMS320 devices can simultaneously provide the multiple functions often required in those complex applications. Table 1–2 lists typical TMS320C1x generation applications.

Table 1–2. Typical Applications of the TMS320C1x Generation

General-Purpose DSP	**Graphics Imaging**	**Instrumentation**
Digital Filtering	3-D Rotation	Spectrum Analysis
Convolution	Robot Vision	Function Generation
Correlation	Image Transm./Comp.	Pattern Matching
Hilbert Transforms	Pattern Recognition	Seismic Processing
Fast Fourier Transforms	Homomorphic Processing	Transient Analysis
Adaptive Filtering	Image Enhancement	Digital Filtering
Windowing	Workstations	Phase-Locked Loops
Waveform Generation	Animation/Digital Map	
Voice/Speech	**Control**	**Military**
Voice Mail	Disk Control	Secure Communications
Speech Vocoding	Servo Control	Radar Processing
Speech Recognition	Robot Control	Sonar Processing
Speaker Verification	Laser Printer Control	Image Processing
Speech Enhancement	Engine Control	Navigation
Speech Synthesis	Motor Control	Missile Guidance
Text-to-Speech		Radio Frequency Modems
Telecommunications		**Automotive**
Echo Cancellation	FAX	Engine Control
ADPCM Transcoders	Cellular Telephones	Vibration Analysis
Digital PBXs	Speaker Phones	Antiskid Brakes
Line Repeaters	X.25 Packet Switching	Adaptive Ride Control
Channel Multiplexing	Video Conferencing	Global Positioning
Adaptive Equalizers	Modems	Voice Commands
DTMF Encoding/Decoding	Data Encryption	Digital Radio
Digital Speech Interpolation	Spread Spectrum Comm.	Navigation
Consumer	**Industrial**	**Medical**
Radar Detectors	Robotics	Hearing Aids
Power Tools	Numeric Control	Patient Monitoring
Digital Audio/TV	Security Access	Ultrasound Equipment
Music Synthesizer	Power Line Monitors	Diagnostic Tools
Educational Toys		Prosthetics
		Fetal Monitors

Chapter 2

Pin Assignments and Signal Descriptions

Electrical specifications, mechanical data, and the signals used in programming the EPROM and one-time programmable devices, are given in the data sheet in Appendix A.

Note:

Throughout this book, C14 designates all devices with a 14 suffix (C14/E14/P14), 'C15 all devices with a 15 suffix (C15/E15/LC15/P15), and 'C17 all devices with a 17 suffix (C17/E17/LC17/P17), unless otherwise noted.

This chapter shows the pin assignments and gives signal descriptions in the following sections:

2.1 TMS320C1x Pin Assignments

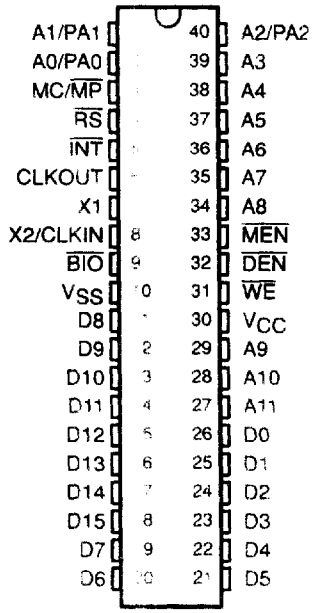
The figures in this section show the pin assignments for TMS320C1x devices.

Refer to the data sheet information in Appendices A through B for the following information:

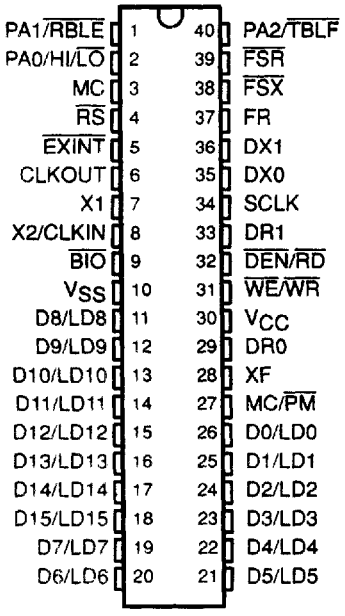
- Electrical specifications
- Mechanical data
- Signals and the procedure used for EPROM programming.

Figure 2–1. TMS320C10, TMS320C15 Series, TMS320C17 Pin Assignments

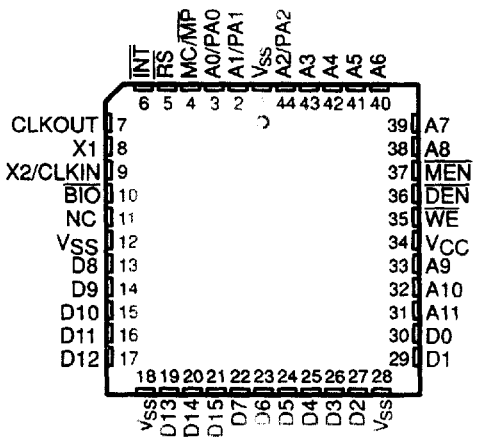
TMS320C10, TMS320C15/LC15, TMS320E15
N (DIP)/JD (CER DIP) Package
(Top View)



TMS320C17/LC17, TMS320E17
N (DIP)/JD (CER DIP) Package
(Top View)



TMS320C10, TMS320C15/LC15, TMS320E15
FN (PLCC) and FZ (CLCC) Packages
(Top View)



TMS320C17/LC17, TMS320E17
FN (PLCC) and FZ (CLCC) Packages
(Top View)

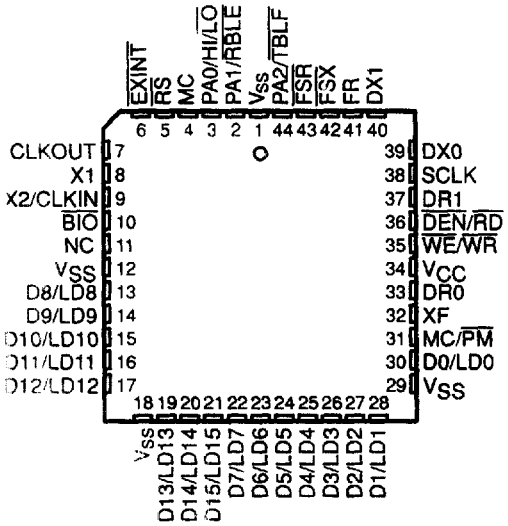


Figure 2–2. TMS320C14 Pin Assignments

TMS320C14, TMS320E14, TMS320P14
FN (PLCC)/FZ (CLCC) Package
(Top View)

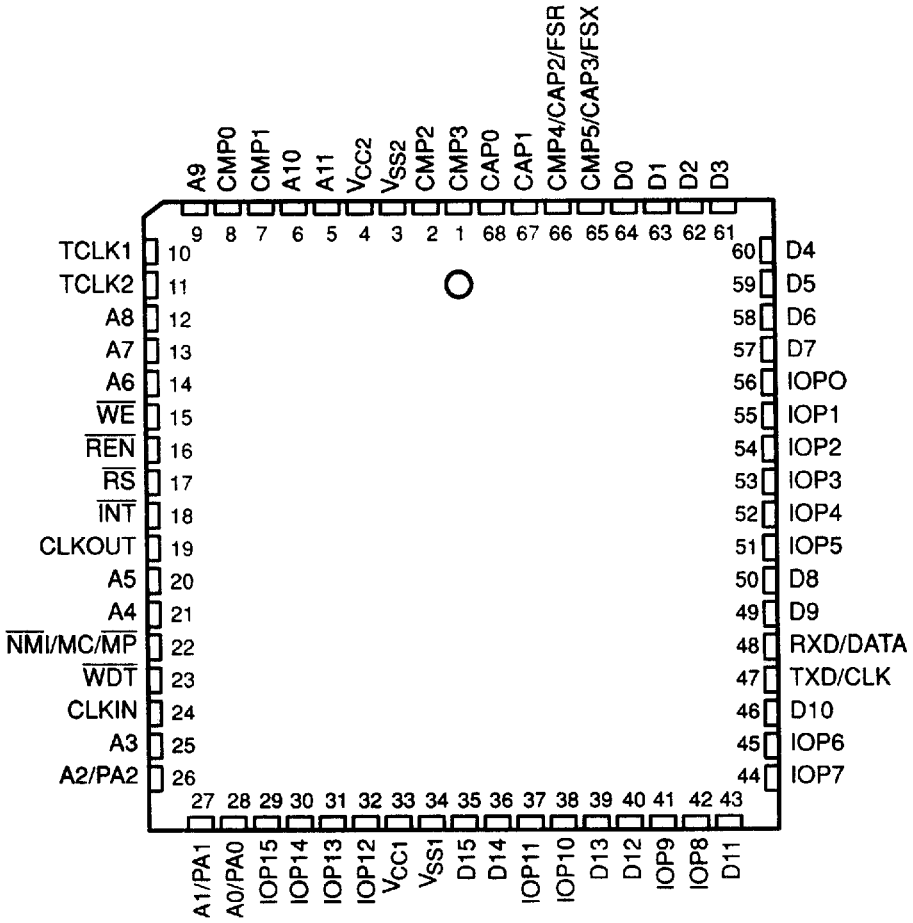
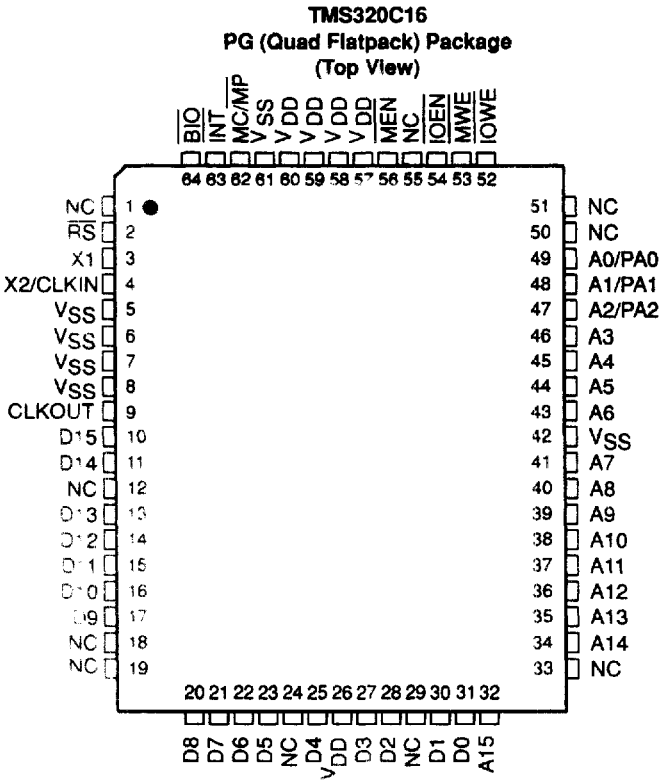


Figure 2–3. TMS320C16 Pin Assignments



2.2 TMS320C10 and TMS320C15 Signal Descriptions

This section gives signal descriptions for the TMS320C10 and TMS320C15 devices. Table 2–1 lists each signal, its pin location according to related package (DIP/PLCC), function, and operating mode(s): that is, input, output, or high-impedance state as indicated by I, O, or Z. The signals in Table 2–1 are grouped according to function and alphabetized within that grouping.

Table 2–1. TMS320C10 and TMS320C15 Signal Descriptions

Signal	Pin DIP/PLCC	I/O/Z†	Description
			Address/Data Buses
A11 MSB A10 A9 A8 A7 A6 A5 A4 A3 A2/PA2 A1/PA1 A0/PA0	27/31 28/32 29/33 34/38 35/39 36/40 37/41 38/42 39/43 40/44 1/2 2/3	O	Program memory address bus A11 (MSB) through A0 (LSB) and port addresses PA2 (MSB) through PA0 (LSB). Addresses A11 through A0 are always active and never go to high impedance. During execution of the IN and OUT instructions, pins A2 through A0 carry the port addresses. (Address pins A11 through A3 are always driven low on IN and OUT instructions
D15 MSB D14 D13 D12 D11 D10 D9 D8 D7 D6 D5 D4 D3 D2 D1 D0 LSB	18/21 17/20 16/19 15/17 14/16 13/15 12/14 11/13 19/22 20/23 21/24 22/25 23/26 24/27 25/29 26/30	I/O/Z	Parallel data bus D15 (MSB) through D0 (LSB). The data bus is always in the high-impedance state except when $\overline{WE}$ is active (low).
			Interrupt and Miscellaneous Signals
$\overline{BIO}$	9/10	I	External polling input. Polled by BIOZ instruction. If low, the device branches to the address specified by the instruction.
$\overline{DEN}$	32/36	O	Data enable for device input data. When active low, $\overline{DEN}$ indicates that the device will accept data from the data bus. $\overline{DEN}$ is active only during the first cycle of the IN instruction. When $\overline{DEN}$ is active, $\overline{MEN}$ and $\overline{WE}$ are always inactive (high).

† Input/output/high-impedance state

Table 2–1. TMS320C10 and TMS320C15 Signal Descriptions (Continued)

Signal	Pin DIP/PLCC	I/O/Z†	Description
Interrupt and Miscellaneous Signals			
$\overline{INT}$	5/6	I	External interrupt input. The interrupt signal is generated by applying a negative-going edge to the $\overline{INT}$ pin. The edge is used to latch the interrupt flag register (INTF) until an interrupt is granted by the device. An active low level is also sensed.
MC/$\overline{MP}$	3/4	I	Memory mode select pin. High selects the microcomputer mode, in which 1.5K words (4K on the TMS320C15/E15) of on-chip program memory are available. This mode also allows an additional 2.5K words of program memory to reside off-chip on the TMS320C10. A low on the MC/$\overline{MP}$ pin enables the microprocessor mode. In this mode, the entire memory space (addresses 0 through 4095) is external.
$\overline{MEN}$	33/37	O	Memory enable. $\overline{MEN}$ is active low on every machine cycle except when $\overline{WE}$ and $\overline{DEN}$ are active. $\overline{MEN}$ is a control signal generated by the device to enable instruction fetches from program memory. $\overline{MEN}$ is active on instructions fetched from both internal and external memory.
NC	/11	–	No connection.
$\overline{RS}$	4/5	I	Reset input for initializing the device. When $\overline{RS}$ is held at an active low for a minimum of five clock cycles, $\overline{DEN}$, $\overline{WE}$, and $\overline{MEN}$ are forced high, and the data bus (D15 through D0) is not driven. The program counter (PC) and the address bus (A11 through A0) are then synchronously cleared after the next complete clock cycle from the falling edge of $\overline{RS}$. Reset also disables the interrupt, clears the interrupt flag register, and leaves the overflow mode register unchanged. The device can be held in the reset state indefinitely.
$\overline{WE}$	31/35	O	Write enable for device output data. When active low, $\overline{WE}$ indicates that the TMS320 will output data onto the data bus. $\overline{WE}$ is active only during the first cycle of the OUT instruction and the second cycle of the TBLW instruction. When $\overline{WE}$ is active, $\overline{MEN}$ and $\overline{DEN}$ are always inactive (high).
Supply and Oscillator Signals			
CLKOUT	6/7	O	System clock output (one-fourth crystal/CLKIN frequency). Duty cycle is fifty percent.
V_{CC}	30/34	I	5-V supply pin.
V_{SS}	10/1, 12, 18, 28	I	Ground pin.
X1	7/8	O	Crystal output pin for internal oscillator. If the internal oscillator is not used, this pin should be left unconnected.
X2/CLKIN	8/9	I	Input pin to the internal oscillator (X2) from the crystal. Alternatively, an input pin for an external oscillator (CLKIN).

2.3 TMS320C14 Signal Descriptions

This section gives signal descriptions for the TMS320C14, TMS320E14, and TMS320P14 devices. Table 2–2 lists each signal, its pin location, operating mode (that is, input, output, high impedance state), and description. The signals are grouped according to function and alphabetized within that group.

Table 2–2. TMS320C14 Signal Descriptions

Signal	Pin PLCC	I/O/Z†	Description
		Address/Data Bus	
A11 MSB A10 A9 A8 A7 A6 A5 A4 A3 A2/PA2 A1/PA1 A0/PA0	5 6 9 12 13 14 20 21 25 26 27 28	O/Z	Program memory address bus A11 (MSB) through A0 (LSB) and port addresses PA2 (MSB) through PA0 (LSB). Addresses A11 through A0 are always active and never go to high impedance except during reset. During execution of the IN and OUT instructions, pins 26, 27, and 28 carry the port addresses. Pins A3 through A11 are held high when port accesses are made on pins PA0 through PA2.
D15 MSB D14 D13 D12 D11 D10 D9 D8 D7 D6 D5 D4 D3 D2 D1 D0 LSB	35 36 39 40 43 46 49 50 57 58 59 60 61 62 63 64	I	Parallel data bus D15 (MSB) through D0 (LSB). The data bus is always in the high-impedance state except when $\overline{WE}$ is active (low). The data bus is also active when internal peripherals are written to.

† Input/Output/high-impedance state

Table 2–2. TMS320C14 Signal Descriptions (Continued)

Signal	Pin PLCC	I/O/Z†	Description
Interrupt and Miscellaneous Signals			
$\overline{INT}$	18	I	External interrupt input. The interrupt signal is generated by a low signal on this pin.
$\overline{NMI}$/MC/$\overline{MP}$	22	I	Non-maskable interrupt. When this pin is brought low, the device is interrupted irrespective of the state of INTM (status register ST) bit. Microcomputer/Microprocessor select. This pin is also sampled when $\overline{RS}$ is low. If $\overline{NMI}$/MC/$\overline{MP}$ is high during reset, internal program memory is selected. If it is low during reset, external program memory is selected.
$\overline{REN}$	16	O	Read enable. When active low, $\overline{REN}$ indicates the device will accept data from the bus.
$\overline{RS}$	17	I	Reset. When this Schmidt trigger input is low, the device is reset and PC is set to zero.
$\overline{WE}$	15	O	Write enable. When active low, $\overline{WE}$ indicates the device will output data on the bus.
Supply/Oscillator Signals			
CLKIN	24	I	Master clock input (from external clock source).
CLKOUT	19	O	System clock output (one fourth CLKIN frequency).
V_{CC}	4, 33	I	5-V supply pins.
V_{SS}	3, 34	I	Ground pins.
Serial Port and Timer Signals			
RXD	48	I/O	Asynchronous mode receive input.
TXD	47	O/Z	Asynchronous mode transmit output.
TCLK1	10	I	Timer 1 clock. If the external clock is selected, it serves as clock input to Timer 1.
TCLK2	11	I	Timer 2 clock. If the external clock is selected, it serves as clock input to Timer 2.
$\overline{WDT}$	23	O	Watchdog timer output. An active low is generated on this pin when the watchdog timer times out.

Table 2–2. TMS320C14 Signal Descriptions (Continued)

Signal	Pin PLCC	I/O/Z†	Description
IOP15 MSB IOP14 IOP13 IOP12 IOP11 IOP10 IOP9 IOP8 IOP7 IOP6 IOP5 IOP4 IOP3 IOP2 IOP1 IOP0 LSB	29 30 31 32 37 38 41 42 44 45 51 52 53 54 55 56	I/O	These are 16-bit I/O lines that can be individually configured as inputs or outputs and also be individually set or reset when configured as outputs.
Compare and Capture Signals			
CMP0 CMP1 CMP2 CMP3	8 7 2 1	O	Compare outputs. The states of these pins are determined by the combination of compare and action registers.
CAP0 CAP1	68 67	I	Capture inputs. A transition on these Schmidt trigger inputs causes the timer register value to be loaded into the corresponding FIFO.
CMP4/CAP2	66	I/O	This pin can be configured as a compare output or input.
CMP5/CAP3	65	I/O	This pin can be configured as a compare output or input.

2.4 TMS320C16 Signal Descriptions

This section gives signal descriptions for the TMS320C16 device. Table 2–3 lists each signal, its pin location, operating mode (that is, input, output, high impedance state), and description. The signals in Table 2–3 are grouped according to function and alphabetized within that group.

Table 2–3. TMS320C16 Signal Descriptions

Signal	Pin QFP	I/O/Z†	Description
			Address/Data Buses
A15 MSB	32	O	Program memory address bus A15 (MSB) through A0 (LSB) and port addresses PA2 (MSB) through PA0 (LSB). Addresses A15 through A0 are always active and never go to high impedance. During execution of the IN and OUT instructions, pins A2 through A0 carry the port addresses. (Address pins A15 through A3 are always driven low on IN and OUT instruction).
A14	34		
A13	35		
A12	36		
A11	37		
A10	38		
A9	39		
A8	40		
A7	41		
A6	43		
A5	44		
A4	45		
A3	46		
A2/PA2	47		
A1/PA1	48		
A0/PA0	49		
D15 MSB	10	I/O/Z	Parallel data bus D15 (MSB) through D0 (LSB). The data bus is always in the high-impedance state except when $\overline{\text{IOWE}}$ or $\overline{\text{MWE}}$ is active (low).
D14	11		
D13	13		
D12	14		
D11	15		
D10	16		
D9	17		
D8	20		
D7	21		
D6	22		
D5	23		
D4	25		
D3	27		
D2	28		
D1	30		
D0 LSB	31		

† Input/output/high-impedance state.

Table 2–3. TMS320C16 Signal Descriptions (Continued)

Signal	Pin QFP	I/O/Z†	Description
Interrupt and Miscellaneous Signals			
$\overline{BIO}$	64	I	External polling input. Polled by BIOZ instruction. If $\overline{BIO}$ is low, the device branches to the address specified by the instruction.
$\overline{IOEN}$	54	O	Data enable for device input data. When active (low), $\overline{IOEN}$ indicates that the device will accept data from the data bus. $\overline{IOEN}$ is active only during the IN instruction. When $\overline{IOEN}$ is active, $\overline{MEN}$, $\overline{IOWE}$, and $\overline{MWE}$ are always inactive (high).
$\overline{IOWE}$	52	O	Write enable for device output data. When active (low), $\overline{IOWE}$ indicates that data will be output from the device on the data bus. $\overline{IOWE}$ is active only during the OUT instruction. When $\overline{IOWE}$ is active, $\overline{MEN}$, $\overline{IOEN}$, and $\overline{MWE}$ are inactive (high).
$\overline{INT}$	63	I	External interrupt input. The interrupt signal is generated by applying a negative-going edge to the $\overline{INT}$ pin. The edge is used to latch the interrupt flag register (INTF) until an interrupt is granted by the device. An active low level will also be sensed.
MC/$\overline{MP}$	62	I	Memory mode select pin. High selects the microcomputer mode, in which 8K words of on-chip program memory are available. A low on the MC/$\overline{MP}$ pin enables the microprocessor mode. In this mode, the entire memory space (addresses 0 through 65535) is external.
$\overline{MEN}$	56	O	Memory enable. $\overline{MEN}$ is an active low control signal generated by the device to enable instruction fetches from program memory. $\overline{MEN}$ is active on instructions fetched from both internal and external memory. When $\overline{MEN}$ is active, $\overline{MWE}$, $\overline{IOWE}$, and $\overline{IOEN}$ will be inactive (high).
$\overline{MWE}$	53	O	Write enable for device output data. When active (low), $\overline{MWE}$ indicates that data will be output from the device on the data bus. $\overline{MWE}$ is active only during the TBLW instruction. When $\overline{MWE}$ is active, $\overline{MEN}$, $\overline{IOEN}$, and $\overline{IOWE}$ are always inactive (high).
NC	1, 12, 18, 19, 24, 29, 33, 50, 51, 55	—	No connection.
$\overline{RS}$	2	I	Schmitt-triggered input for initializing the device. When $\overline{RS}$ is held active for a minimum of five clock cycles, $\overline{IOEN}$, $\overline{IOWE}$, $\overline{MWE}$, and $\overline{MEN}$ are forced high, and the data bus (D15 through D0) is not driven. The program counter (PC) and the address bus (A15 through A0) are then synchronously cleared after the next complete clock cycle from the falling edge of $\overline{RS}$. Reset also disables the interrupt, clears the interrupt flag register, and leaves the overflow mode register unchanged. The device can be held in the reset state indefinitely.
Supply/Oscillator Signals			
CLKOUT	9	O	System clock output (one-fourth crystal/CLKIN frequency).

Table 2–3. TMS320C16 Signal Descriptions (Continued)

Signal	Pin QFP	I/O/Z†	Description
V_{DD}	26, 57, 58, 59, 60	I	5-V supply pins.
V_{SS}	5, 6, 7, 8, 42, 61	I	Ground pins.
X1	3	O	Crystal output pin for internal oscillator. If the internal oscillator is not used, this pin should be left unconnected.
X2/CLKIN	4	I	Input pin to the internal oscillator (X2) from the crystal. Alternatively, an input pin for an external oscillator (CLKIN).

† Input/output/high-impedance state

2.5 TMS320C17 Signal Descriptions

Table 2–4 lists each signal provided on the TMS320C17, its pin location, function, and operating mode(s), that is, input, output, or high-impedance state as indicated by I, O, or Z. The signals in Table 2–4 are grouped according to function and alphabetized within that grouping. Note that the first signal and the signal following the slash are both used on the TMS320C17/E17.

Table 2–4. TMS320C17 Signal Descriptions

Signal	Pin DIP/PLCC	I/O/Z†	Description
Bidirectional Data Bus			
D15/LD15 D14/LD14 D13/LD13 D12/LD12 D11/LD11 D10/LD10 D9/LD9 D8/LD8 D7/LD7 D6/LD6 D5/LD5 D4/LD4 D3/LD3 D2/LD2 D1/LD1 D0/LD0	18/21 17/20 16/19 15/17 14/16 13/15 12/14 11/13 19/22 20/23 21/24 22/25 23/26 24/27 25/28 26/30	I/O/Z	In the microcomputer mode, these data lines represent a 16-bit parallel data bus (D15 through D0). The data bus is always in the high-impedance state, except when $\overline{WE}$ is active (low) or when an IN instruction is being executed from either port 0 or port 1. In the coprocessor mode, the 16-bit data lines (LD15 through LD0) are used for a coprocessor latch. The data bus is always held in a high-impedance state except when $\overline{RD}$ is active (low).
Port Address Bus			
PA2/$\overline{TBLF}$ PA1/$\overline{RBLE}$ PA0/HI/$\overline{LO}$	40/44 1/2 2/3	O O I/O/Z	I/O port address output/transmit buffer latch full flag. I/O port address output/receive buffer latch empty flag. I/O port address output/latch byte select pin. In the microcomputer mode, these pins carry the port address when IN and OUT instructions are used. When other instruction cycles are used, these pins carry the three LSBs of the program counter. In the coprocessor mode, these pins signal the status of the receive and the transmit buffer latches.

† Input/output/high-impedance state

Table 2–4. TMS320C17 Signal Descriptions (Continued)

Signal	Pin DIP/PLCC	I/O/Z†	Description
Interrupt and Miscellaneous Signals			
$\overline{BIO}$	9/10	I	External polling input. Polled by the BIOZ instruction. If $\overline{BIO}$ is low, the device branches to the address specified by the instruction. When the device is in the coprocessor mode, the $\overline{BIO}$ line is reserved for coprocessor interface and cannot be driven externally.
$\overline{DEN}/\overline{RD}$	32/36	I/O/Z	Data enable for device input data/external read for the output latch. When active low, $\overline{DEN}$ indicates that the device accepts data from the data bus. $\overline{DEN}$ is active only during the first cycle of the IN instruction. $\overline{WE}$ is always inactive (high) when $\overline{DEN}$ is active. In the coprocessor mode, the external processor reads from the coprocessor latch by driving the $\overline{RD}$ line active (low), thus enabling the output latch to drive the latched data. When the data has been read, the external device brings the $\overline{RD}$ line high.
$\overline{EXINT}$	5/6	I	External interrupt input. The interrupt signal is generated by applying a logic low level to the $\overline{EXINT}$ pin.The edge is used to latch the system control register flag bit (CR0) until an interrupt is granted by the device. In the coprocessor mode, the $\overline{EXINT}$ line is reserved for coprocessor interface and cannot be driven externally.
MC	3/4	I	Microcomputer mode select pin. The MC pin must be connected to the same state as the MC/$\overline{PM}$ pin. When these pins are low, the coprocessor port is enabled. When these pins are high, the microcomputer mode is enabled
MC/$\overline{PM}$	27/31	I	Microcomputer or peripheral/coprocessor mode select pin.This pin must be connected to the same state as the MC pin. When these pins are low, the coprocessor port is enabled. When these pins are high, the microcomputer mode is enabled.
NC	/11	–	No connection
$\overline{RS}$	4/5	I	Reset input for initializing the device. When an active low is placed on the $\overline{RS}$ pin for a minimum of five clock cycles, both $\overline{DEN}$ and $\overline{WE}$ are forced high, and the data bus (D15 through D0) goes to a high-impedance state. The serial port clock and transmit outputs also go to the high-impedance state. The program counter (PC) and the port address bus (PA2 through PA0) are then synchronously cleared after the next complete clock cycle from the falling edge of $\overline{RS}$.
$\overline{WE}/\overline{WR}$	31/35	I/O	Write enable for device output data/external write enable for the input latch. When active low, $\overline{WE}$ indicates that the TMS320 will output data onto the data bus. $\overline{WE}$ is active only during the first cycle of the OUT instruction and the second cycle of the TBLW instruction. $\overline{DEN}$ is always inactive (high) when $\overline{WE}$ is active. In the coprocessor mode, the external processor lowers the $\overline{WR}$ line and places data on the bus. It next raises the $\overline{WR}$ line to clock the data into the on-chip latch.
XF	28/32	O	External logic output flag. Programmable via system control register bit 10 (CR10) This pin is the direct output of the CR10 latch.

Table 2–4. TMS320C17 Signal Descriptions (Continued)

Signal	Pin DIP/PLCC	I/O/Z†	Description
			Supply/Oscillator Signals
CLKOUT	6/7	O	System clock output (one-fourth crystal/CLKIN frequency).
V_{CC}	30/34	I	5-V supply pin.
V_{SS}	10/1, 12, 18, 29	I	Ground pin.
X1	7/8	O	Crystal output pin for internal oscillator. If the internal oscillator is not used, this pin should be left unconnected.
X2/CLKIN	8/9	I	Input pin to the internal oscillator (X2) from the crystal. Alternatively, an input pin for an external oscillator (CLKIN).
			Serial Port Signals
DR1 DR0	33/37 29/33	I	Serial-port receive-channel inputs. Serial data is received in the internal receive registers via these pins.
DX1 DX0	36/40 35/39	O/Z	Serial-port transmit-channel outputs. Serial data is transmitted from the transmit registers onto these pins of the processor. These outputs are in the high-impedance state when not transmitting.
FR	37/41	O	Internal serial-port framing output. If internal framing is enabled, serial-port transmit and receive operations occur simultaneously on an active (high) FR framing pulse. Both short and long FR pulses are selectable to provide fixed and variable data-rate framing pulses for combo-codec interface. The FR frequency is derived from the serial-port clock (SCLK) and system control register bits CR23–CR16.
$\overline{FSR}$	39/43	I	External serial-port receive-framing input. If external framing is enabled via the system control register, data is received via the receive pins (DR1 and DR0) on the active (low) $\overline{FSR}$ input. The falling edge of $\overline{FSR}$ initiates the receive process, and the rising edge sets the flag bit (CR1) in the system control register, causing an interrupt to occur if enabled.
$\overline{FSX}$	38/42	I	External serial-port transmit-framing input. If external framing is enabled, data is transmitted on the transmit pins (DX1,DX0) on the active (low) $\overline{FSX}$ input. The falling edge of $\overline{FSX}$ initiates the transmit process, and the rising edge sets the flag bit (CR2) in the system control register, causing an interrupt to occur if enabled.
SCLK	34/38	I/O/Z	Serial-port clock. Master clock for transmitting and receiving serial-port data. Configurable as an input or output. SCLK must always be present for serial-port operation. As an input, SCLK is the external clock that controls data transfers with the serial port. As an output, SCLK provides the serial clock for data transfers and framing-pulse synchronization. Its frequency is derived from the TMS320C17/E17 system clock, X2/CLKIN, and from system control register bits CR27–CR24. Reset ($\overline{RS}$) forces SCLK to the high-impedance state.

† Input/output/high-impedance state

Chapter 3

TMS320C1x Architecture

The hardware-intensive design of the TMS320C1x devices provides performance previously unavailable on a single chip. Hardware implements functions that other processors typically perform in software. For example, the hardware multiplier performs the multiplication process during one instruction cycle. Flexibility is further enhanced by the comprehensive instruction set, which supports either general-purpose or digital signal processing applications.

In a strict Harvard architecture, program and data memory lie in two separate spaces, permitting full overlap of instruction fetch and execution. TheTMS320C1x modification of the Harvard architecture allows transfers between program and data spaces, increasing device speed and flexibility. This modification permits coefficients stored in program memory to be read into RAM, eliminating the need for a separate coefficient ROM. It also makes available immediate instructions and subroutines based on computed values.

Note:
Throughout this book, 'C14 designates all devices with a 14 suffix (C14/E14/P14), 'C15 all devices with a 15 suffix (C15/E15/LC15/P15), and 'C17 all devices with a 17 suffix (C17/E17/LC17/P17), unless otherwise noted.

Major topics in this chapter are:

3.1 Architectural Overview

The TMS320C1x devices contain a 32-bit ALU and accumulator for supporting double-precision, 2s-complement arithmetic. The ALU is a general-purpose arithmetic unit; operations are performed with the 16-bit words taken from data RAM, the 16-bit words derived from immediate instructions, or the 32-bit result taken from the product register of the multiplier. In addition to the usual arithmetic instructions, the ALU can perform Boolean operations, providing the bit manipulation ability required of a high-speed controller. The accumulator stores the output from the ALU and is often an input to the ALU. The accumulator is 32 bits long and is divided into a high-order word (bits 31–16) and a low-order word (bits 15–0). Instructions are provided for storing the high- and low-order accumulator words in memory.

The multiplier performs a 16 × 16-bit 2s-complement multiplication with a 32-bit result in a single instruction cycle. The multiplier consists of three elements:

- the T Register,
- the P Register, and
- the multiplier array.

The 16-bit T register temporarily stores the multiplicand; the P register stores the 32-bit product. Multiplier values either come from the data memory or are derived immediately from the MPYK (multiply immediate) instruction word. The fast on-chip multiplier allows the device to perform fundamental DSP operations such as convolution, correlation, and filtering.

Two shifters are available for manipulating data. The ALU barrel shifter performs a left-shift of 0 to 16 places on data memory words loaded into the ALU. This shifter extends the high-order bit of the data word and zero-fills the low-order bits for 2s-complement arithmetic. The accumulator parallel shifter performs a left-shift of 0, 1, or 4 places on the entire accumulator and stores the resulting high-order accumulator bits into data RAM. Both shifters are useful for scaling and bit extraction.

3.1.1 On-Chip Memory

The TMS320C1x devices have 144/256 words of on-chip data RAM and 1.5K/4K words of on-chip program ROM/EPROM to support program development. The EPROM cell utilizes standard PROM programmers and is programmed the same way as a 64K CMOS EPROM (TMS27C64). The TMS320C1x devices can execute programs from up to 4K words of off-chip memory at full speed for those applications requiring external program memory space. This allows external RAM-based systems to provide multiple functionality. The TMS320C17 does not provide memory expansion capability.

3.1.2 Modes of Operation

The TMS320C10 and TMS320C15 devices offer two modes of operation defined by the state of the MC/$\overline{\text{MP}}$ pin: the microcomputer mode (high level) or

the microprocessor mode (low level). In the microcomputer mode, on-chip ROM is mapped into the memory space with up to 4K words of memory available. In the microprocessor mode, all 4K words of memory are external.

The $\overline{NMI}$/MC/$\overline{MP}$ pin on the TMS320C14 also selects the microcomputer or microprocessor mode. However, the pin is sampled for mode selection while $\overline{RS}$ is low. Otherwise, the pin is used as a nonmaskable interrupt.

3.1.3 Hardware Stack

The TMS320C1x devices (except the 'C16) contain a four-level hardware stack for saving the contents of the program counter during interrupts and subroutine calls. Special instructions can save the device's complete context. PUSH and POP instructions permit a level of nesting restricted only by the amount of available RAM. The interrupts are maskable.

On the TMS320C16 the depth of the hardware stack has been increased to 8. This increase in nesting ability is in response to the 'C16's ability to access a larger (64K) program space.

3.1.4 I/O Ports

The 16-bit parallel data bus can perform I/O functions in two cycles. The I/O ports are addressed by the three LSBs on the address lines. In addition, a polling input for bit test and branch operations ($\overline{BIO}$) and an interrupt pin ($\overline{INT}$) increase system flexibility. Two of the I/O ports on the TMS320C17 are dedicated to the serial port and companding hardware. I/O port 0 is dedicated to control register 0, which controls the serial port, interrupts, and companding hardware. I/O port 1 accesses control register 1, as well as both serial port channels, and the companding hardware. The six remaining I/O ports are available for external parallel interfaces. On the TMS320C17, port 5 can be used for coprocessor interface. When port 5 is used, ports 2, 3, 4, 6, and 7 are not accessible.

The TMS320C17 offers a dual-channel serial port capable of full-duplex serial communication and direct interface to combo-codecs. Receive and transmit registers that operate with 8-bit data samples are I/O-mapped. Either internal or external framing signals for serial data transfers are selected through the system control register. The serial port clock furnishes the bit timing for transfers with the serial port and may be either an input or an output. A framing pulse signal supplies framing pulses for combo-codec circuits, an 8-kHz sample clock for voice-band systems, or a timer for control applications.

3.1.5 Data Companding (TMS320C17)

On-chip hardware can compand (COMpress/exPAND) data in either μ-law (U.S. and Japan) or A-law (European) format. The companding logic operation

is configured via the system control register. Data may be companded in either a serial mode for operation on serial port data (converting between linear and logarithmic PCM) or in a parallel mode for computation inside the device. The TMS320C17 allows the hardware companding logic to operate with either sign-magnitude or 2s-complement numbers.

3.1.6 Coprocessor Interfacing

The coprocessor port on the TMS320C17 connects directly to most 4-/8-bit microcomputers and 16-/32-bit microprocessors. In the coprocessor mode, the 16-bit parallel port is reconfigured to operate as a 16-bit latched bus interface. Data widths of either 8 or 16 bits may be selected for the coprocessor port, accessed through I/O port 5 using IN and OUT instructions. The coprocessor interface allows the device to act as a peripheral (slave) microcomputer to a microprocessor or as a master to a peripheral microcomputer. In the microcomputer mode, the 16 data lines are used for the six parallel 16-bit I/O ports.

3.2 Block Diagrams and Internal Hardware Summaries

Figure 3–1 through Figure 3–4 show the block diagrams with the main blocks and data paths of the TMS320C1x devices. Table 3–1 through Table 3–4 describe alphabetically the internal hardware shown in the figures.

Figure 3–1. TMS320C10 and TMS320C15 Block Diagram

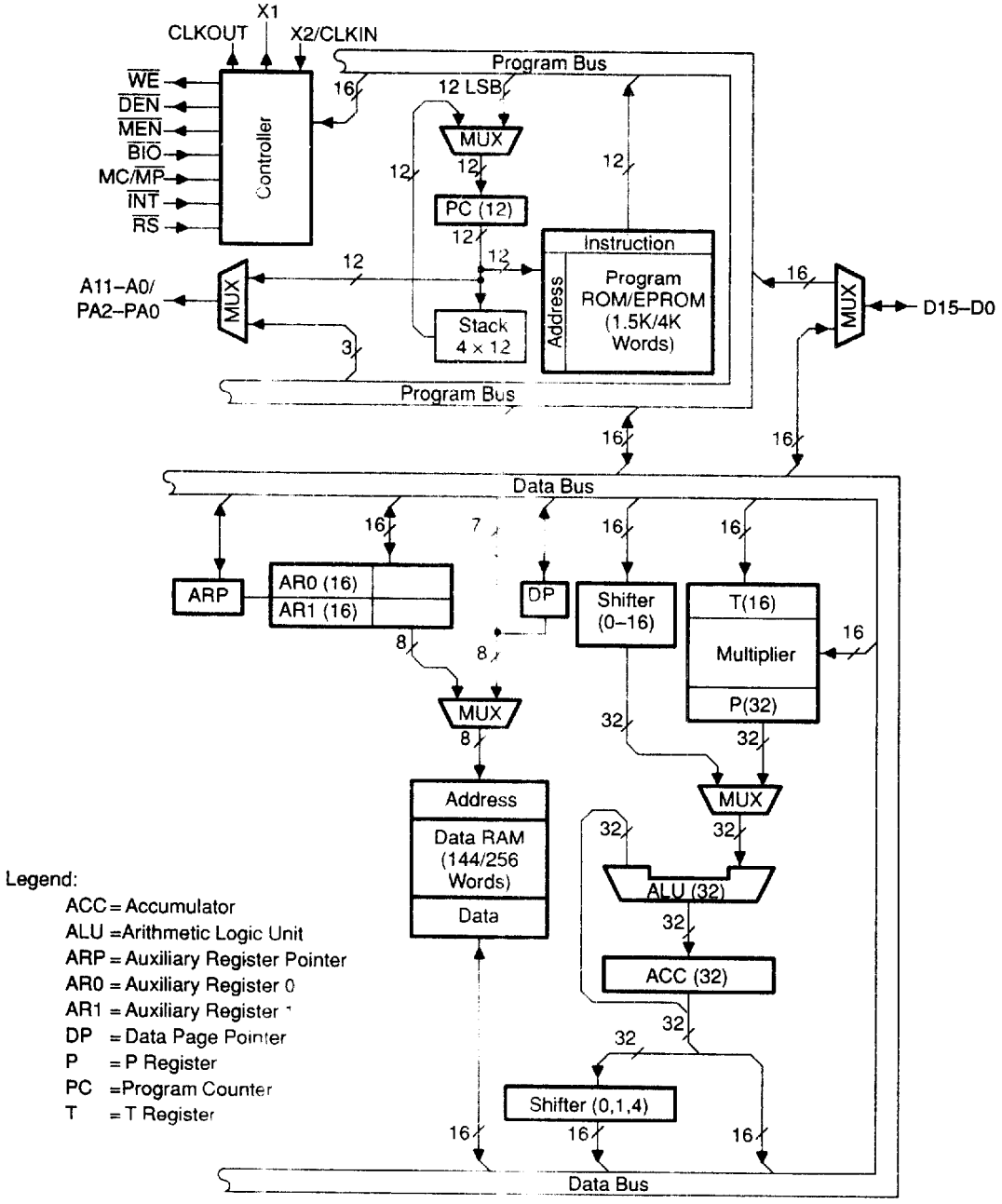
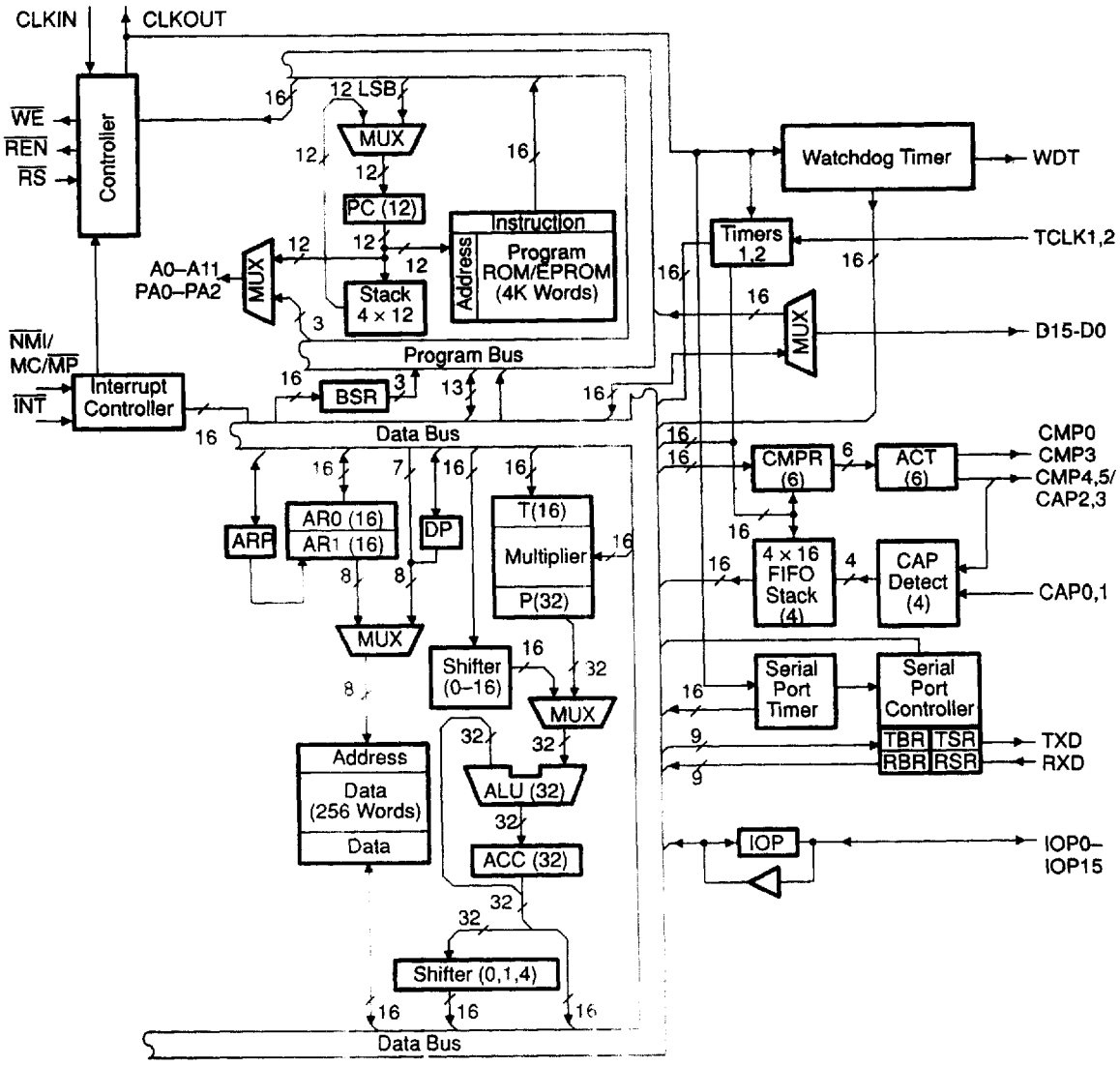


Table 3–1. TMS320C10/C15 Internal Hardware

Unit	Symbol	Function
Accumulator	ACC	A 32-bit accumulator divided into a high-order word (bits 31 through 16) and a low-order word (bits 15 through 0). Used for storage of ALU output.
Arithmetic Logic Unit	ALU	A 32-bit 2s-complement arithmetic logic unit with two 32-bit input ports and one 32-bit output port feeding the accumulator.
Auxiliary Registers	AR0, AR1	Two 16-bit registers used for data memory addressing and loop count control. Nine LSBs of each register are configured as up/down counters.
Auxiliary Register Pointer	ARP	A status bit that indicates the currently active auxiliary register.
Central Arithmetic Logic Unit	CALU	The grouping of the ALU, multiplier, accumulator, and shifters.
Data Bus	D(15–0)	A 16-bit bus used to route data to and from RAM.
Data Memory Page Pointer	DP	A status bit that points to the data RAM address of the current page. A data page contains 128 words.
Data RAM	–	144 or 256 words of on-chip random access memory containing data.
External Address Bus	A(11–0)/PA(2–0)	A 12-bit bus used to address external program memory. The three LSBs are port addresses in the I/O mode.
Interrupt Flag	INTF	A single-bit flag that indicates an interrupt request has occurred (is pending).
Interrupt Mode	INTM	A status bit that masks the interrupt flag.
Multiplier	MULT	A 16 × 16-bit parallel hardware multiplier.
Overflow Flag	OV	A status bit flag that indicates an overflow in arithmetic operations.
Overflow Mode	OVM	A status bit that defines a saturated or unsaturated mode in arithmetic operations.
P Register	P	A 32-bit register containing the product of multiply operations.
Program Bus	P(15–0)	A 16-bit bus used to route instructions from program memory.
Program Counter	PC(11–0)	A 12-bit register used to address program memory. The PC always contains the address of the next instruction to be executed. The PC contents are updated following each instruction decode operation.
Program ROM/EPROM	–	1.5K or 4K words of on-chip read-only memory (ROM or EPROM) containing the program code.
Shifters	–	Two shifters: the ALU barrel shifter that performs a left-shift of 0 to 16 bits on data memory words loaded into the ALU, and the accumulator parallel shifter that performs a left-shift of 0, 1, or 4 places on the entire accumulator and places the resulting high-order bits into data RAM.
Stack	–	Memory used to store the PC during interrupts or calls.
Status Register	ST	A16-bit status register that contains status and control bits.
T Register	T	A16-bit register containing the multiplicand during multiply operations.

Figure 3–2. TMS320C14 Block Diagram



Legend:

ACC	=	Accumulator	DP	=	Data Page Pointer
ACT	=	Action Register (Bit Selectable	IOP	=	Input/Output Port
ALU	=	Arithmetic Logic Unit	P	=	P Register
ARP	=	Auxiliary Register Pointer	PC	=	Program Counter
AR0	=	Auxiliary Register 0	RBR	=	Receive Buffer Register
AR1	=	Auxiliary Register 1	RSR	=	Receive Shift Register
BSR	=	Bank Select Register	T	=	T Register
CAP	=	Capture	TBR	=	Transmit Buffer Register
CMPR	=	Compare Register	TSR	=	Transmit Shift Register

Table 3–2. TMS320C14 Internal Hardware

Unit	Symbol	Function
Accumulator	ACC	A 32-bit accumulator divided into a high-order word (bits 31 through 16) and a low-order word (bits 15 through 0). Used for storage of ALU output.
Arithmetic Logic Unit	ALU	A 32-bit 2s-complement arithmetic logic unit with two 32-bit input ports and one 32-bit output port feeding the accumulator.
Auxiliary Registers	AR0, AR1	Two 16-bit registers used for data memory addressing and loop count control. Nine LSBs of each register are configured as up/down counters.
Auxiliary Register Pointer	ARP	A status bit that indicates the currently active auxiliary register.
Central Arithmetic Logic Unit	CALU	The grouping of the ALU, multiplier, accumulator, and shifters.
Data Bus	D(15–0)	A 16-bit bus used to route data to and from RAM.
Data Memory Page Pointer	DP	A status bit that points to the data RAM address of the current page. A data page contains 128 words.
Data RAM	–	256 words of on-chip random access memory containing data.
External Address Bus	A(11–0)/PA(2–0)	A 12-bit bus used to address external program memory. The three LSBs are port addresses in the I/O mode.
Interrupt Flag	INTF	A single-bit flag that indicates an interrupt request has occurred (is pending).
Interrupt Mode	INTM	A status bit that masks the interrupt flag.
Multiplier	MULT	A 16 × 16-bit parallel hardware multiplier.
Overflow Flag	OV	A status bit flag that indicates an overflow in arithmetic operations.
Overflow Mode	OVM	A status bit that defines a saturated or unsaturated mode in arithmetic operations.
P Register	P	A 32-bit register containing the product of multiply operations.
Program Bus	P(15–0)	A 16-bit bus used to route instructions from program memory.
Program Counter	PC(11–0)	A 12-bit register used to address program memory. The PC always contains the address of the next instruction to be executed. The PC contents are updated following each instruction decode operation.
Program ROM/EPROM	–	4K words of on-chip read-only memory (ROM or EPROM) containing the program code.
Shifters	–	Two shifters: the ALU barrel shifter that performs a left-shift of 0 to 16 bits on data memory words loaded into the ALU, and the accumulator parallel shifter that performs a left-shift of 0, 1, or 4 places on the entire accumulator and places the resulting high-order bits into data RAM.
Stack	–	A 4 × 12-bit memory used to store the PC during interrupts or calls.
Status Register	ST	A16-bit status register that contains status and control bits.
T Register	T	A16-bit register containing the multiplicand during multiply operations.

Figure 3–3. TMS320C16 Block Diagram

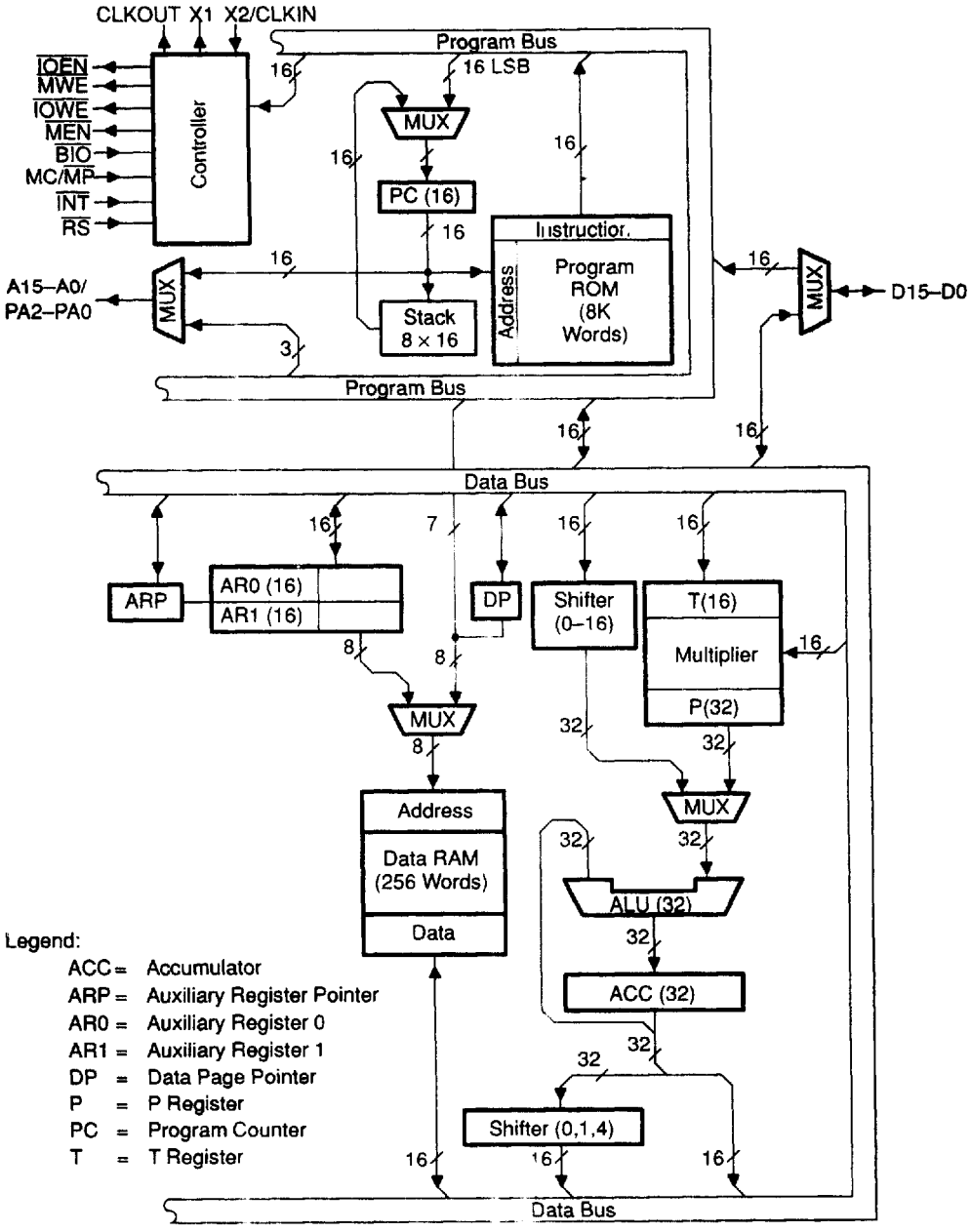
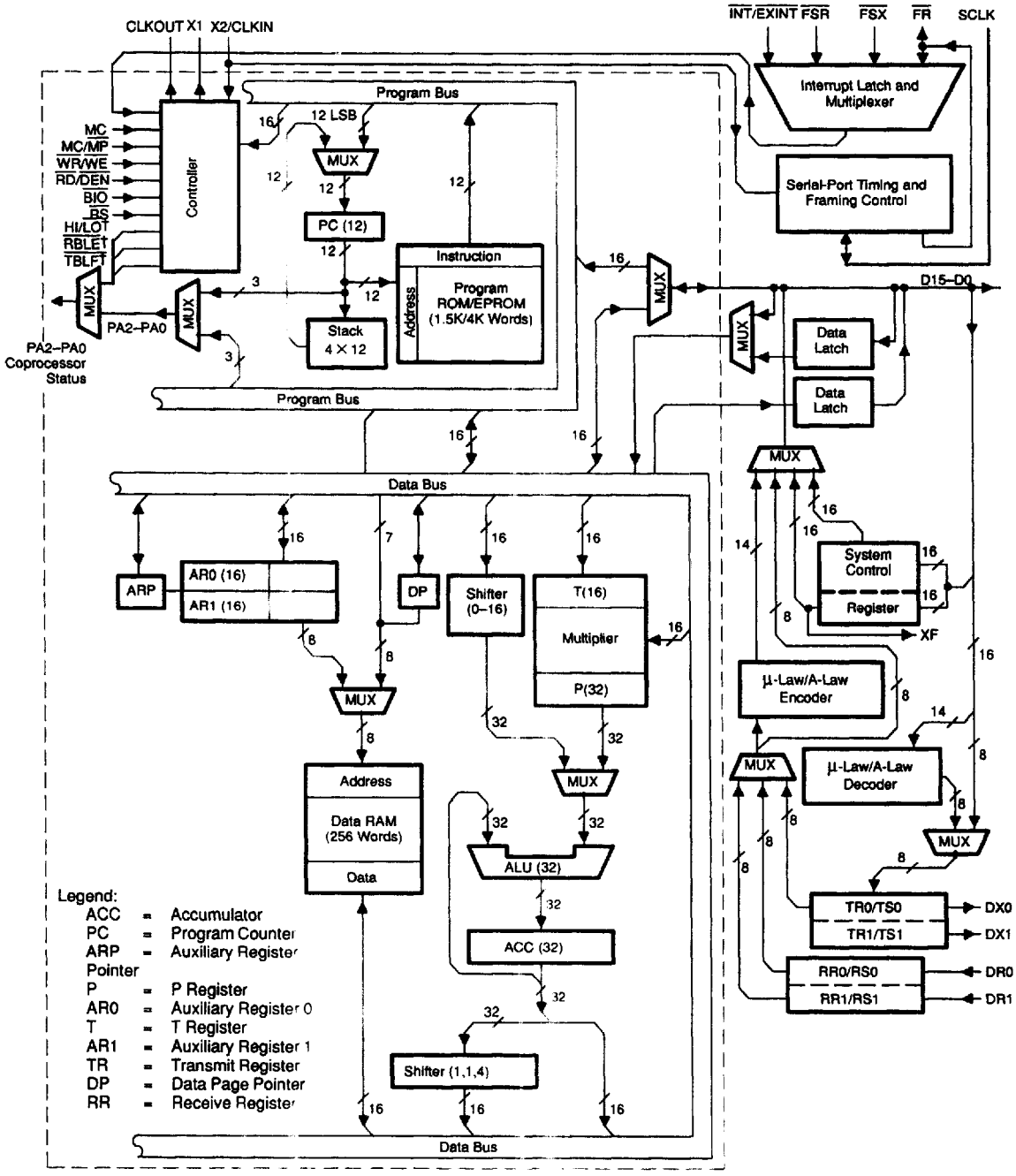


Table 3–3. TMS320C16 Internal Hardware

Unit	Symbol	Function
Accumulator	ACC	A 32-bit accumulator divided into a high-order word (bits 31 through 16) and a low-order word (bits 15 through 0). Used for storage of ALU output.
Arithmetic Logic Unit	ALU	A 32-bit 2s-complement arithmetic logic unit with two 32-bit input ports and one 32-bit output port feeding the accumulator.
Auxiliary Registers	AR0, AR1	Two 16-bit registers used for data memory addressing and loop count control. Nine LSBs of each register are configured as up/down counters.
Auxiliary Register Pointer	ARP	A status bit that indicates the currently active auxiliary register.
Central Arithmetic Logic Unit	CALU	The grouping of the ALU, multiplier, accumulator, and shifters.
Data Bus	D(15–0)	A 16-bit bus used to route data to and from RAM.
Data Memory Page Pointer	DP	A status bit that points to the data RAM address of the current page. A data page contains 128 words.
Data RAM	–	256 words of on-chip random access memory containing data.
External Address Bus	A(15–0)/PA(2–0)	A 16-bit bus used to address external program memory. The three LSBs are port addresses in the I/O mode.
Interrupt Flag	INTF	A single-bit flag that indicates an interrupt request has occurred (is pending).
Interrupt Mode	INTM	A status bit that masks the interrupt flag.
Multiplier	MULT	A 16 × 16-bit parallel hardware multiplier.
Overflow Flag	OV	A status bit flag that indicates an overflow in arithmetic operations.
Overflow Mode	OVM	A status bit that defines a saturated or unsaturated mode in arithmetic operations.
P Register	P	A 32-bit register containing the product of multiply operations.
Program Bus	P(15–0)	A 16-bit bus used to route instructions from program memory.
Program Counter	PC(11–0)	A 16-bit register used to address program memory. The PC always contains the address of the next instruction to be executed. The PC contents are updated following each instruction decode operation.
Program ROM	–	8K words of on-chip read-only memory (ROM) containing the program code.
Shifters	–	Two shifters: the ALU barrel shifter that performs a left-shift of 0 to 16 bits on data memory words loaded into the ALU, and the accumulator parallel shifter that performs a left-shift of 0, 1, or 4 places on the entire accumulator and places the resulting high-order bits into data RAM.
Stack	–	An 8 × 16-bit memory used to store the PC during interrupts or calls.
Status Register	ST	A16-bit status register that contains status and control bits.
T Register	T	A16-bit register containing the multiplicand during multiply operations.

Figure 3–4. TMS320C17 Block Diagram



TMS320C15 Core CPU (ASIC Library)

Table 3–4. TMS320C17 Internal Hardware

Unit	Symbol	Function
Accumulator	ACC	A 32-bit accumulator divided into a high-order word (bits 31 through 16) and a low-order word (bits 15 through 0). Used for storage of ALU output.
Arithmetic Logic Unit	ALU	A 32-bit 2s-complement arithmetic logic unit with two 32-bit input ports and one 32-bit output port feeding the accumulator.
Auxiliary Registers	AR0, AR1	Two 16-bit registers used for data memory addressing and loop count control. Nine LSBs of each register are configured as up/down counters.
Auxiliary Register Pointer	ARP	A status bit that indicates the currently active auxiliary register.
Central Arithmetic Logic Unit	CALU	The grouping of the ALU, multiplier, accumulator, and shifters.
Companding Hardware	–	Data companding encoder/decoder in either μ-law or A-law PCM conversion format. Two modes of operation: serial mode for operating on serial port data (linear/logarithmic PCM conversions) or parallel mode for computation inside the device. Companding is selected through the control register.
Data Bus	D(15–0)	A 16-bit bus used to route data to and from RAM.
Data Memory Page Pointer	DP	A status bit that points to the data RAM address of the current page. A data page contains 128 words.
Data RAM	–	256 words of on-chip random access memory containing data.
External Address Bus	A(11–0)/PA(2–0)	A 12-bit bus used to address external program memory. The three LSBs are port addresses in the I/O mode.
Interrupt Flag	INTF	A single-bit flag that indicates an interrupt request has occurred (is pending).
Interrupt Mode	INTM	A status bit that masks the interrupt flag.
Multiplier	MULT	A 16 × 16-bit parallel hardware multiplier.
Latched Data Bus	LD(15–0)	A 16-bit bidirectional latched data bus used in coprocessor mode. This bus is connected internally to two latches, one for input and one for output.
Overflow Flag	OV	A status bit flag that indicates an overflow in arithmetic operations.
Overflow Mode	OVM	A status bit that defines a saturated or unsaturated mode in arithmetic operations.
P Register	P	A 32-bit register containing the product of multiply operations.
Program Bus	P(15–0)	A 16-bit bus used to route instructions from program memory.
Program Counter	PC(11–0)	A 12-bit register used to address program memory. The PC always contains the address of the next instruction to be executed. The PC contents are updated following each instruction decode operation.
Program ROM/EPROM	–	4K words of on-chip read-only memory (ROM or EPROM) containing the program code.
Serial Port Clock	SCLK	The clock that provides the timing control for data transfers with the serial port. SCLK is configured through the control register.

Table 3–4. TMS320C17 Internal Hardware (Continued)

Unit	Symbol	Function
Serial Port Framing Control	FR	A signal that provides serial port framing compatible with combo-codec devices. The FR pulse signifies a transmit/receive of new data on the serial port.
Serial Port Receive Registers	RR0,RR1	8-bit serial port registers that receive 8-bit data samples.
Serial Port Receive Shift Registers	RS0,RS1	8-bit registers used to shift in serial port data from pin DR0 or DR1.
Serial Port Transmit Registers	TR0,TR1	8-bit serial port transmit registers in a FIFO (first in, first out) configuration.
Serial Port Transmit Shift Registers	TS0,TS1	8-bit registers used to shift out serial port data onto pin DX0 or DX1.
Shifters	–	Two shifters: the ALU barrel shifter that performs a left-shift of 0 to 16 bits on data memory words loaded into the ALU, and the accumulator parallel shifter that performs a left-shift of 0, 1, or 4 places on the entire accumulator and places the resulting high-order bits into data RAM.
Stack	–	Memory used to store the PC during interrupts or calls.
Status Register	ST	A16-bit status register that contains status and control bits.
System Control Register	CR(31–0)	A 32-bit register that controls interrupts, serial port channels, companding hardware, and coprocessor port channels. Control register 1, accessed through port 1, consists of the upper 16 bits (CR31–CR16). Control register 0, accessed through port 0, consists of the lower 16 bits (CR15–CR0).
T Register	–	A16-bit register containing the multiplicand during multiply operations.

3.3 Memory Organization

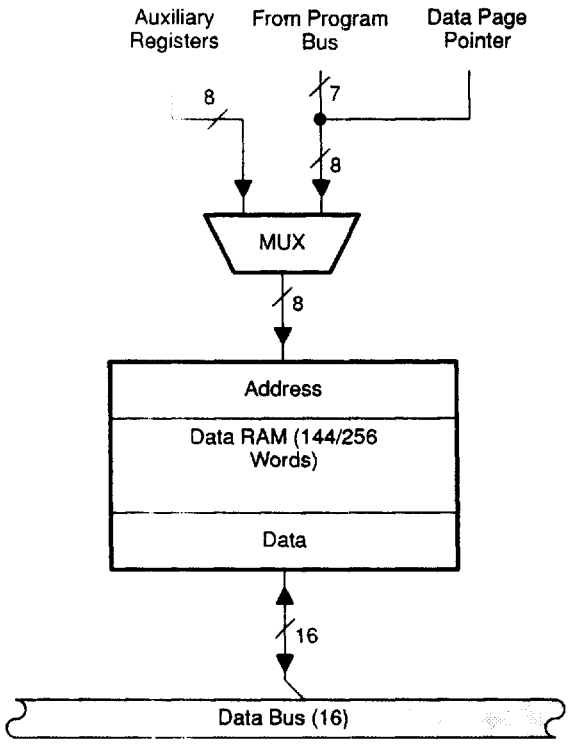
The TMS320C1x devices use a Harvard architecture, in which data and program memory reside in two separate spaces. The TMS320C1x provides 144/256 16-bit words of on-chip data RAM and 1.5K/4K words of program ROM. On-chip program EPROM versions are available. Also, the TMS320C16 has an internal 8K of ROM and can access 64K of external program space. This section describes the TMS320C1x data and program memory, data movement, memory maps, auxiliary registers, microcomputer/microprocessor modes, and memory addressing modes.

3.3.1 Data Memory

Data memory consists of 144/256 words of 16-bit on-chip RAM (see Figure 3–5). The TMS320C10 provides 144 words. The TMS320C14, TMS320C15, TMS320C16, and TMS320C17 have expanded on-chip RAM of 256 words. See subsection 3.3.4 for memory map configurations.

To expand data memory, the data operands may be stored off-chip and then read into the on-chip RAM as they are needed. Two instruction pairs, TBLR/TBLW and IN/OUT, do this. The table read (TBLR) instruction can transfer values from program memory, (either on-chip ROM or off-chip ROM/RAM), to the on-chip data RAM. The table write (TBLW) instruction transfers values from the data RAM to off-chip program RAM. These instructions execute in three cycles. When you use the IN/OUT instruction pair, the IN instruction reads data from a peripheral and transfers it to the data RAM. With some extra hardware, the IN instruction, together with the OUT instruction, can read and write from the data RAM to large amounts of external storage addressed as a peripheral. This method is faster because the IN and OUT instructions execute in only two cycles. See Section 6.1 for hardware applications using RAM/ROM expansion.

Figure 3–5. On-Chip Data Memory



3.3.2 Program Memory

TMS320C1x program memory consists of up to 8K internal words and up to 64K external words. The program memory for the different devices is as follows

Device	Program Memory (ROM)
TMS320C10	1.5K words
TMS320C15 and 'C17	4K words
TMS320C16	8K words

The TMS320C16 offers the largest internal ROM of 8K. This on-chip program ROM allows program execution at full speed without the need for high-speed external program memory. On-chip program EPROM of 4K words, provided on the TMS320E15 and TMS320E17, presents two additional benefits. First, direct programming of the EPROM greatly facilitates application development. Second, these devices implement a security feature that can be used to protect proprietary algorithms by preventing the EPROM contents from being read.

Program memory operation is user-selectable by means of the MC/$\overline{\text{MP}}$ (microcomputer/microprocessor) pin. Setting MC/$\overline{\text{MP}}$ high places the device in the

microcomputer mode. Holding the pin low places the device in the microprocessor mode.

In the microcomputer mode, an internal ROM is mapped into the program space of all TMS320C1x devices. When the program address is beyond the upper memory boundary of the ROM, the external program bus is utilized. For devices such as the TMS320C15, whose internal ROM is the same size (4K) as the total program reach, external program access is not possible while in this mode. On other devices such as the TMS320C10, the internal ROM is not as large as the total program space and the external bus becomes active in this region

One important aspect of microcomputer mode operation is that the internal ROM is tested at the factory by using a small area of it. This area is at the highest address and is reserved by TI for all ROM coded devices. Note that this section of reserved ROM will create a hole in program code that goes through this area. The reserved ROM area should be avoided when converting programs from all external to mixed internal/external memory. The following table and the memory maps in subsection 3.3.4 show the program memory configuration while in microcomputer mode.

Figure 3–6. Microcomputer Mode Program Memory Allocation

Device	Program Space	ROM Space	Reserved ROM	External
TMS320C10	0000–0FFFh	0000–0FFFh	05F4–05FFh	0600–0FFFh
TMS320C14	0000–0FFFh	0000–0FFFh	0FA0h–0FFFh	None
TMS320C15	0000–0FFFh	0000–0FFFh	0FA0h–0FFFh	None
TMS320C16	0000–FFFFh	0000–1FFFh	1FB0h–1FFFh	2000–FFFFh
TMS320C17	0000–0FFFh	0000–0FFFh	0FA0h–0FFFh	None

For those applications requiring external program memory, interfacing is provided by the address and memory control pins. However, note that the interface is not identical among devices, and that the TMS320C17 has no external program memory.

External RAM or ROM can be interfaced to the TMS320C1x (see Section 6.1) for those applications requiring external program memory space. This creates multiple functionality for external RAM-based systems. The TMS320C17 provides no direct program memory expansion capability.

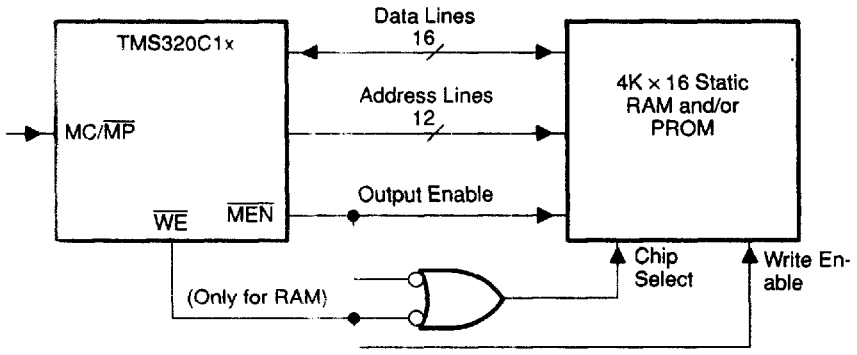
For the TMS320C10, 'C14, and 'C15, twelve output pins are available for addressing external memory. These pins, A11 (MSB) through A0 (LSB), contain the buffered outputs of the program counter or the I/O port address. When an instruction is fetched from off-chip memory, the $\overline{\text{MEN}}$ (memory enable) strobe is generated to enable the external memory. The instruction word is then transferred to the processor via the data bus (see Section 3.6).

When in the microcomputer mode, the processor selects internal program memory. The $\overline{\text{MEN}}$ strobe still becomes active in this mode, and the address

lines A11 through A0 still output the current value of the program counter, although the instruction word is read from internal program memory. Note that $\overline{MEN}$ is never active at the same time as the $\overline{WE}$ or $\overline{DEN}$ signals. In effect, $\overline{MEN}$ goes low every clock cycle except when an I/O function is being performed by the IN, OUT, or TBLW instructions. In these multicycle instructions, $\overline{MEN}$ goes low during the clock cycles in which $\overline{WE}$ or $\overline{DEN}$ do not go low.

Figure 3–7 gives an example of external program memory expansion. Even when executing from external memory, the TMS320C1x performs at full speed. Note that some ports are reserved for on-chip peripheral logic.

Figure 3–7. External Program Memory Expansion Example



3.3.3 Data Movement

Data movement functions within on-chip RAM can be performed with the DMOV and LTD instructions. These instructions are useful for implementing algorithms that use the z^{-1} delay operation, such as convolutions and digital filtering, where data is passed through a time window.

Implemented in on-chip RAM, these instructions allow a word to be copied from the currently addressed data memory location in on-chip RAM to the next higher location while the data from the addressed location is being operated upon in the same cycle (for example, by the CALU).

3.3.4 Memory Maps

The TMS320C1x devices provide three separate address spaces for program memory, data memory, and I/O, as shown in Figure 3–8 through Figure 3–11. Program memory is configured according to the state of the MC/$\overline{MP}$ pin. For further information about data and program memory, see subsections 3.3.1 and 3.3.2. I/O functions are discussed in Section 3.6.

Figure 3–8. Memory Maps for the TMS320C10

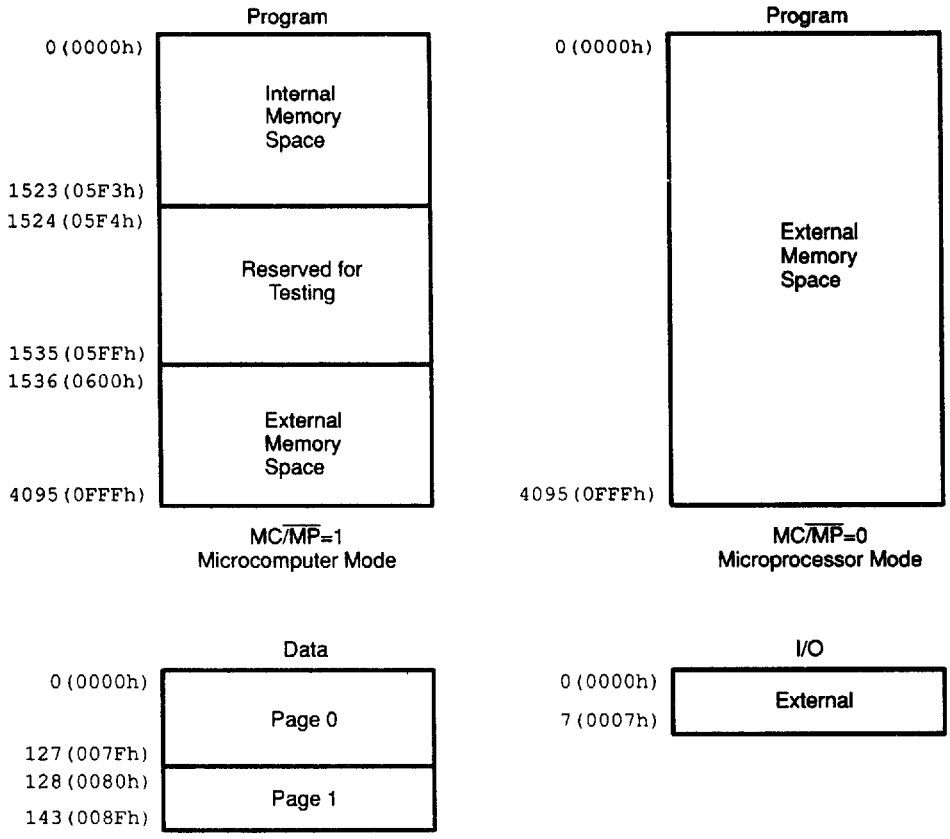


Figure 3–9. Data Memory Map for the TMS320C14

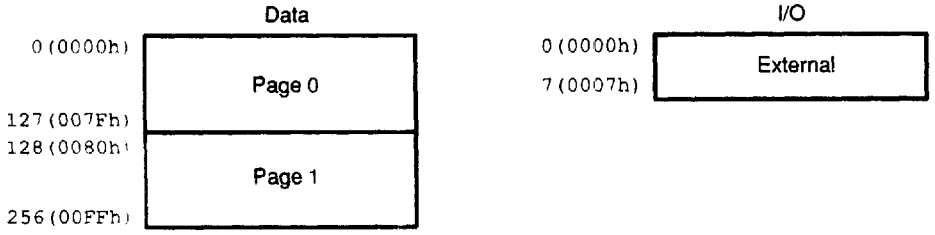
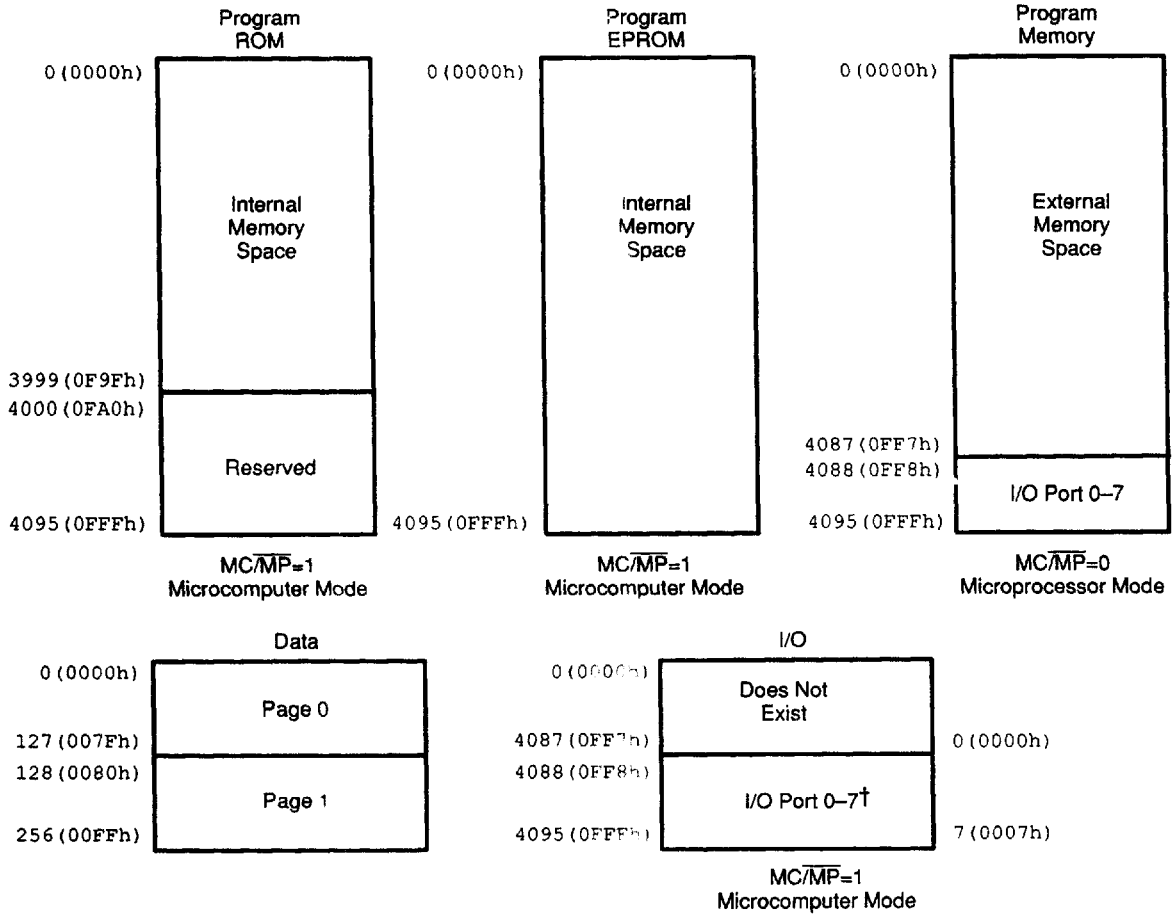


Figure 3–10. Program Memory Map for the TMS320C14



† During in and out instructions address lines A3–A11 are driven high. This is true for both microcomputer and microprocessor modes. Since $\overline{RE}$ and $\overline{WE}$ are active for all external signals the I/O ports have been mapped to the highest words of the external program space to avoid program memory conflicts. If external program is not used PA0–PA2 can be used to decode I/O space.

Figure 3–11. Memory Maps for the TMS320C15

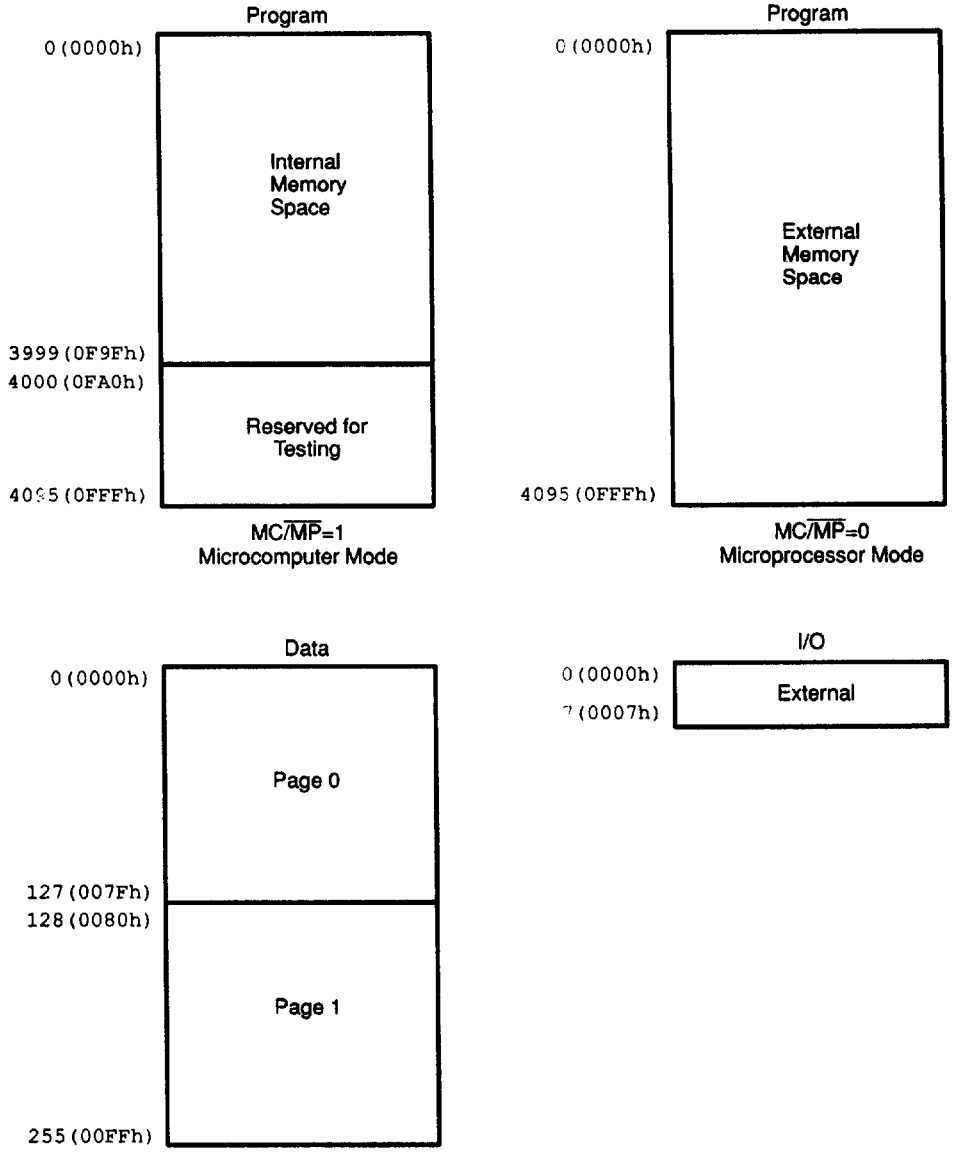


Figure 3–12. Memory Maps for the TMS320C16

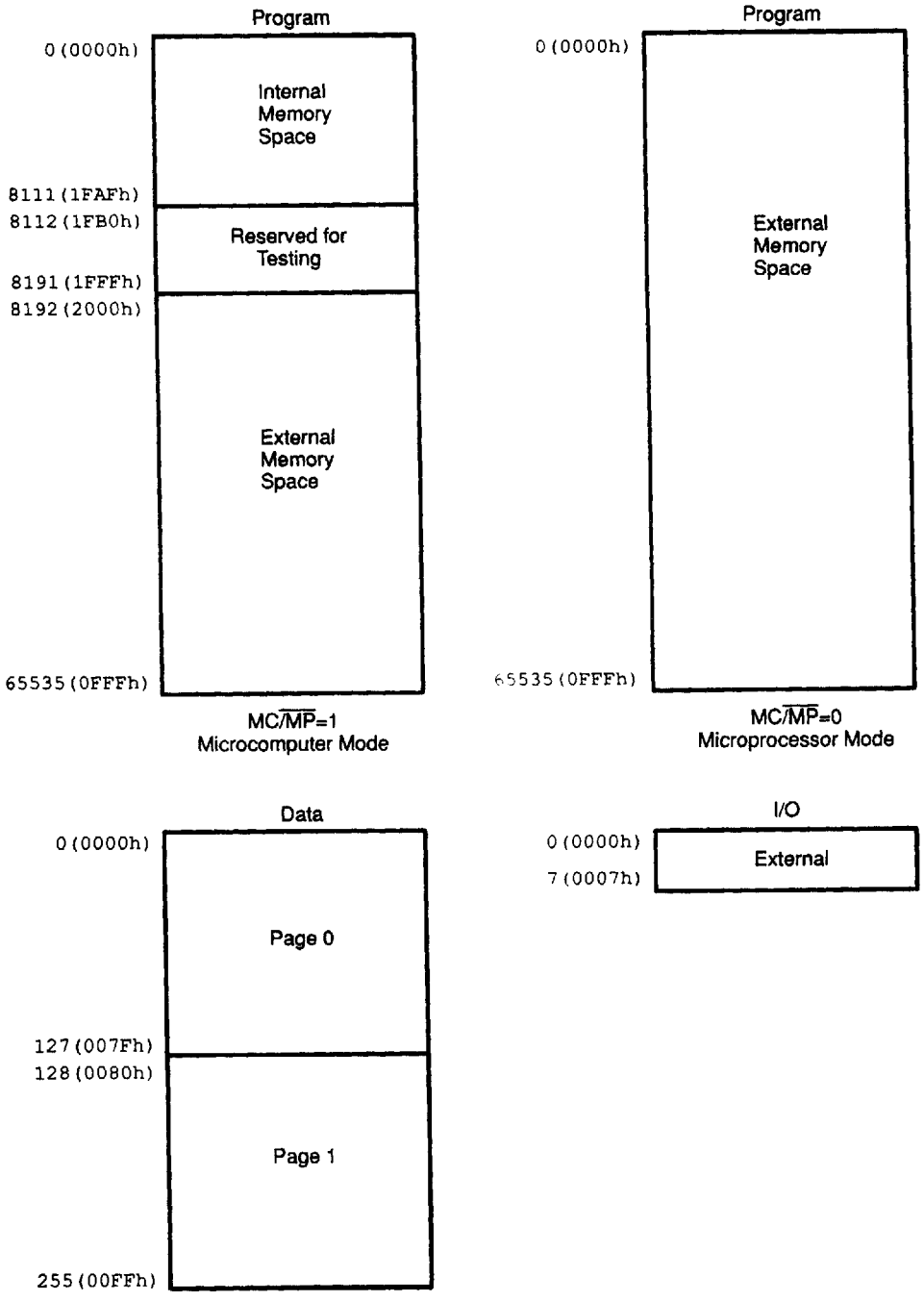
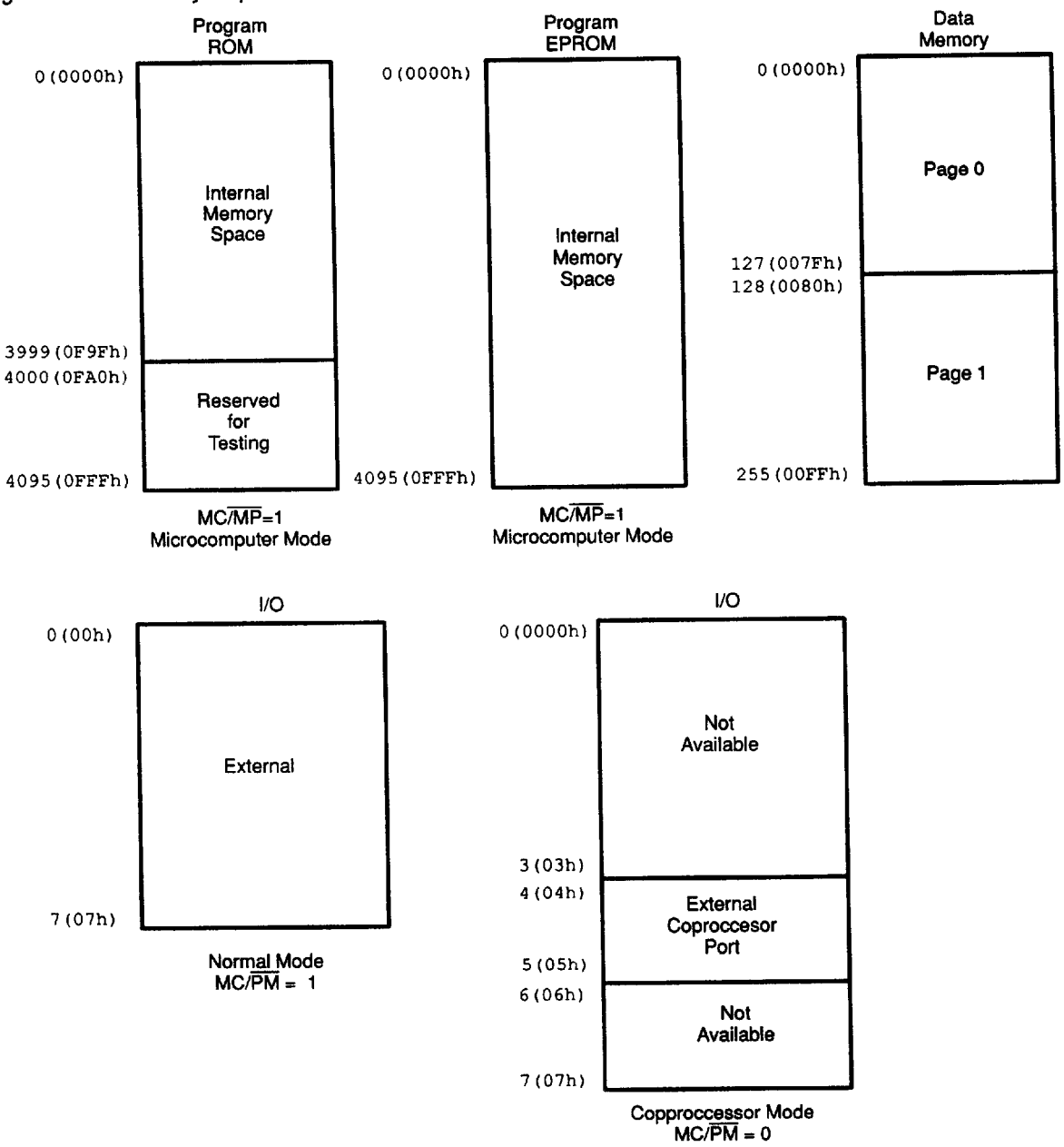


Figure 3–13. Memory Map for TMS320C17



Note: The TMS320C17 does not have any means of directly accessing an external data space.

3.3.5 Auxiliary Registers

The TMS320C1x devices provide two 16-bit auxiliary registers (AR0 and AR1). This section discusses each register's function and how an auxiliary register is selected, loaded, and stored.

The auxiliary registers may be used for indirect addressing of data memory, temporary data storage, and loop control. Indirect addressing allows placement of the data memory address of an instruction operand into the least significant eight bits of an auxiliary register. The registers are selected by a single-bit auxiliary register pointer (ARP) that is loaded with a value of 0 or 1, designating AR0 or AR1, respectively. The ARP is part of the status register and can be stored in memory.

When the auxiliary registers are autoincremented/decremented by an indirect addressing instruction or by the BANZ (branch on auxiliary register not zero) instruction, the lowest nine bits are affected (see Figure 3–14). The auxiliary registers can be used as loop counters when the BANZ instruction is used. This counter portion of an auxiliary register is a 9-bit counter, as shown in Figure 3–15 and Figure 3–16.

The upper seven bits of an auxiliary register (that is, bits 9 through 15) are unaffected by any autoincrement/decrement operation. This includes autoincrement of 111111111 (the lowest nine bits go to 0) and autodecrement of 000000000 (the lowest nine bits go to 111111111); in each case, bits 9 through 15 are unaffected.

The auxiliary registers can be saved in and loaded from data memory with the SAR (store auxiliary register) and LAR (load auxiliary register) instructions. This is useful for performing context saves. SAR and LAR transfer entire 16-bit values to and from the auxiliary registers even though indirect addressing and loop counting use only a portion of the auxiliary register. See Section 4.1.2 for programming of the indirect addressing mode.

The BANZ instruction checks whether the lower 9 bits of an auxiliary register are zero. If not, it decrements and branches. See subsection 5.3.3 for loop code using the auxiliary registers.

Figure 3–14. Auxiliary Register Counter

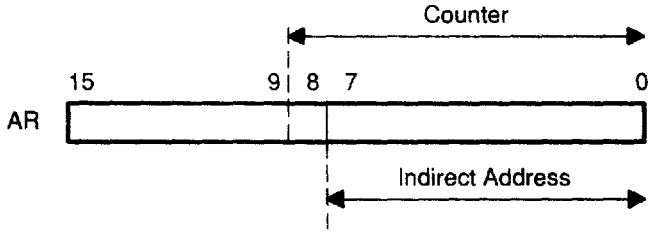


Figure 3–15. Indirect Addressing Autoincrement

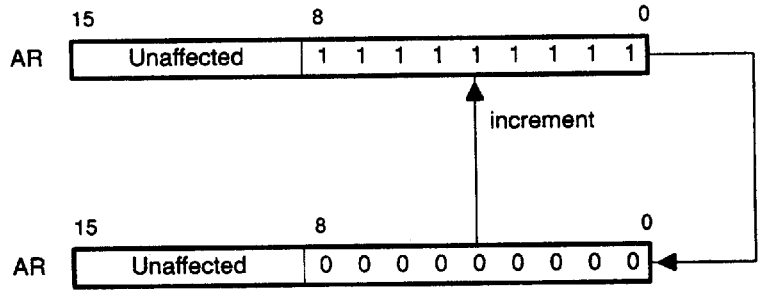
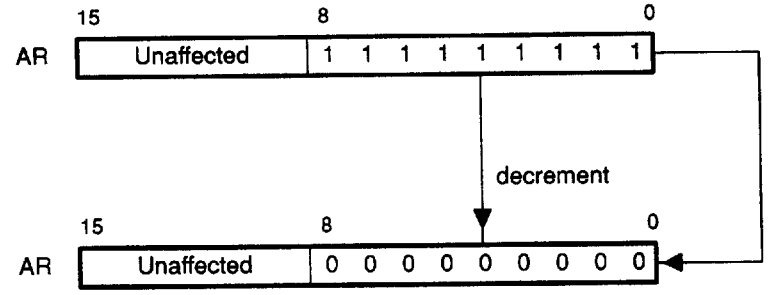


Figure 3–16. Indirect Addressing Autodecrement



3.3.6 Addressing Modes

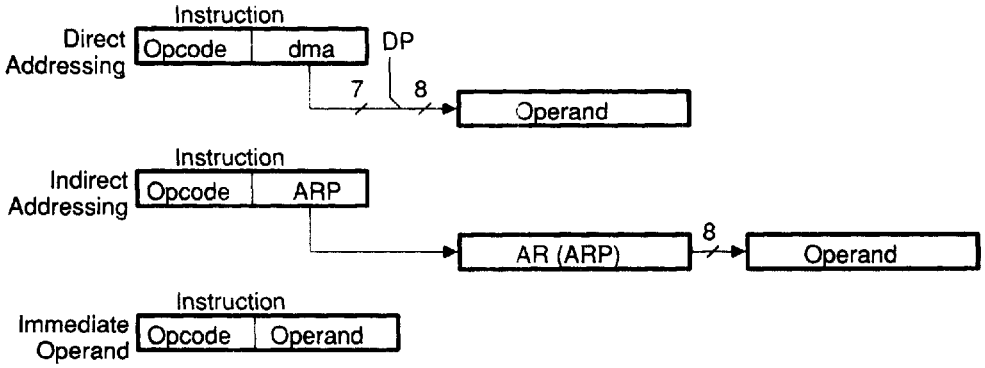
Three forms of instruction operand addressing can be used: direct, indirect, and immediate. Figure 3–17 illustrates operand addressing in these three modes, which are described in detail in Section 4.1.

In the direct addressing mode, the 1-bit data memory page pointer (DP) selects either page 0, consisting of memory locations 0–127, or page 1, consisting of locations 128–143/255. The data memory address (dma), specified by the seven LSBs of the instruction concatenated with the DP, addresses the desired word within the page. Note that DP is part of the status register and thus can be stored in data memory.

Indirect addressing uses the lower eight bits of the auxiliary registers as the data memory address. This is sufficient to address all 256 data words; no paging is necessary with indirect addressing. The current auxiliary register is selected by the auxiliary register pointer (ARP). In addition, the auxiliary registers can be made to autoincrement/decrement during any given indirect instruction. Note that the increment/decrement occurs after the current instruction is finished executing.

When an immediate operand is used, the instruction word contains the operand itself.

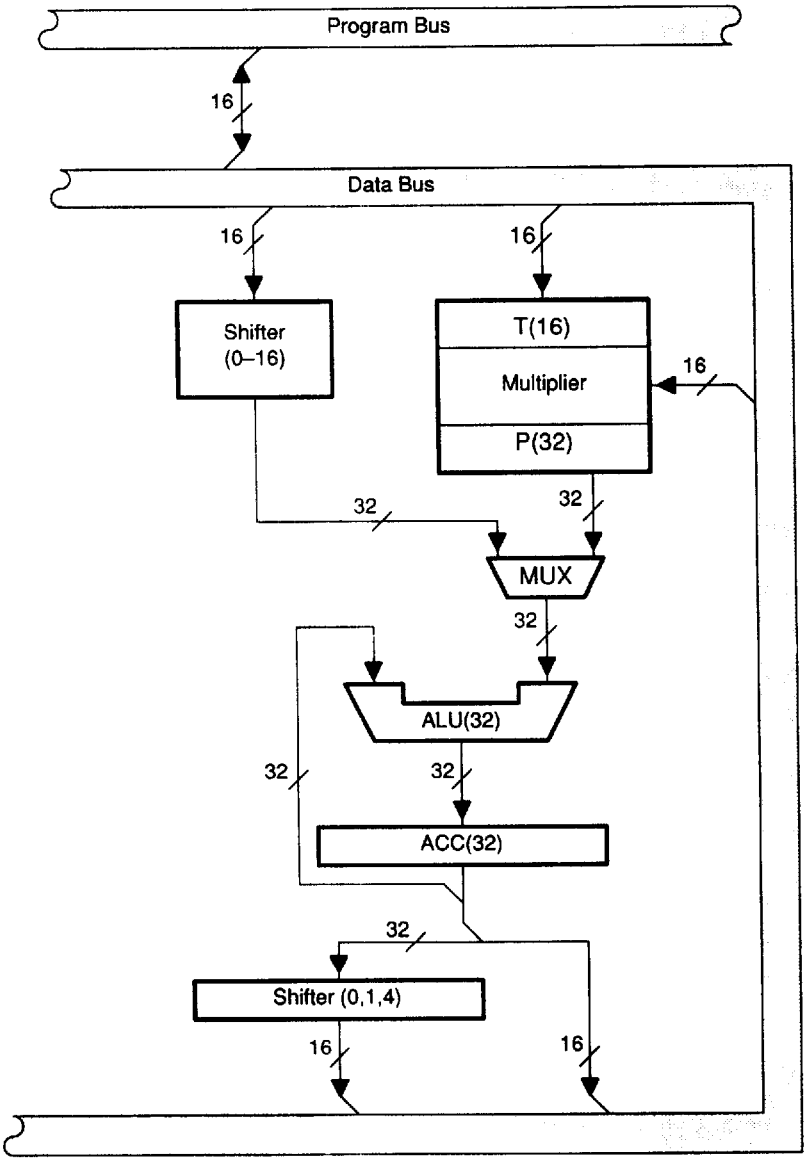
Figure 3–17. Methods of Instruction Operand Addressing



3.4 Central Arithmetic Logic Unit (CALU)

The central arithmetic logic unit (CALU) contains a 16 × 16-bit parallel multiplier, a 32-bit arithmetic logic unit (ALU), a 32-bit accumulator (ACC), and two shifters. This section describes the CALU components and their functions. Figure 3–18 is a block diagram showing the components of the CALU.

Figure 3–18. Central Arithmetic Logic Unit (CALU)



The following steps occur in the implementation of a typical ALU operation:

1) Data is fetched from the RAM on the data bus.

2) Data is passed through the barrel shifter where it can be left-shifted 0 to16 bits, depending on the value specified by the instruction.

3) Data enters the ALU where it is operated upon and loaded into the accumulator.

4) The result obtained in the accumulator is passed through a parallel left-shifter present at the accumulator output to aid in scaling results.

5) The result is stored in the data RAM. Because the accumulator is 32 bits wide, each half must be stored separately.

One input to the ALU is always provided from the accumulator, and the other input may be provided from the P Register of the multiplier or the barrel shifter that is loaded from data memory.

3.4.1 Shifters

Two shifters are available for manipulating data: a barrel shifter for shifting data from the data RAM into the ALU and a parallel shifter for shifting the accumulator into the data RAM (see Figure 3–18).

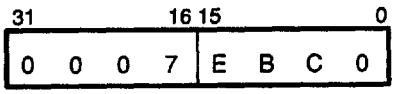
The barrel shifter has a 16-bit input connected to the data bus and a 32-bit output connected to the ALU. The barrel shifter produces a left shift of 0 to 16 bits on all data memory words that are loaded into, subtracted from, or added to the accumulator by the LAC, SUB, and ADD instructions. The shifter zero-fills the LSBs and sign-extends the 16-bit data memory word to 32 bits by an arithmetic left shift (that is, the bits to the left of the MSB of the data word are filled with ones if the MSB is a one or with zeros if the MSB is a zero). This differs from a logical left shift in which the bits to the left of the MSB are always filled with zeros. A small amount of code is required to perform an arithmetic right-shift or a logical right-shift.

The following examples illustrate the barrel shifter's function:

- Data memory location 20 holds the 2s-complement number 7EBCh. The LAC (load accumulator) instruction is executed, specifying a left shift of 4:

```
LAC 20,4
```

- The accumulator then holds the following 32-bit signed 2s-complement number:



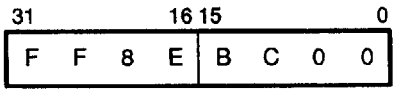
Because the MSB of 7EBCh is a zero, the upper accumulator was zero-filled.

- Data memory location 30 holds the 2s-complement number: 8EBCh.

  The LAC (load accumulator) instruction is executed, specifying a left- shift of 8:

```
LAC 30,8
```

- The accumulator then holds the following 32-bit signed 2s-complement number:

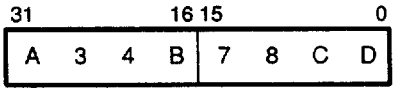


  Because the MSB of 8EBCh is a one, the upper accumulator was filled with ones.

Certain instructions perform operations with the lower half of the accumulator and a data word without first sign-extending the data word (that is, treating it as a 16-bit rather than as a 32-bit word). The mnemonics of these instructions typically end with an S, indicating that sign-extension is suppressed (for example, ADDS, SUBS). Along with the instructions that operate on the upper half of the accumulator, these instructions allow the manipulation of 32-bit precision numbers

The parallel shifter is activated only by the SACH (store high-order accumulator word) instruction. This instruction causes the shifter to be loaded with the 32-bit contents of the accumulator. The data is then left-shifted. The most significant 16 bits from the shifter are stored in RAM, resulting in a loss of the high-order bits of data. The contents of the accumulator remain unchanged. The parallel shifter can execute a shift of only 0,1, or 4. Shifts of 1 and 4 are used with multiplication operations. No right shift is directly implemented. The following example illustrates the accumulator shifter's function:

- The accumulator holds the following 32-bit signed 2s-complement number:



  The SACH instruction is executed, specifying that a left shift of 4 be performed on the high-order accumulator word before it is stored in data memory location 40:

```
SACH 40,4
```

  Data memory location 40 then contains the 2s-complement number 34B7h. The accumulator retains 0A34B78CDh.

3.4.2 ALU and Accumulator

The 32-bit ALU and accumulator (see Figure 3–18) implement a wide range of arithmetic and logical functions, the majority of which execute in a single clock cycle. Once an operation is performed in the ALU, the result is transferred to the accumulator where additional operations such as shifting may occur. Data that is input to the ALU may be scaled by the barrel shifter.

The ALU is a general-purpose arithmetic logic unit that operates on 16-bit data words, producing a 32-bit result. The ALU can add, subtract, and perform logical operations. The accumulator is always the destination and the primary operand. The result of logical operations is shown in Table 3–5. A data memory address (dma) is the operand for the lower half of the accumulator (bits 15 through 0). Zero is the operand for the upper half of the accumulator.

Table 3–5. Accumulator Results of a Logical Operation

Function	ACC Bits 31–16	ACC Bits 15–0
XOR	(0).XOR.(ACC (31–16))	(dma).XOR.(ACC (15–0))
AND	(0).AND.(ACC (31–16))	(dma).AND.(ACC (15–0))
OR	(0).OR.(ACC (31–16))	(dma).OR.(ACC (15–0)

The 32-bit accumulator stores the output from the ALU and is also often an input to the ALU. The accumulator is divided into two 16-bit words for storage in data memory: a high-order word (bits 31 through 16) and a low-order word (bits 15 through 0). The SACH and SACL instructions are used to store the high- and low-order accumulator words, respectively, in data memory. These instructions can be used in the implementation of double-precision arithmetic.

A shifter at the output of the accumulator provides a left shift of 0, 1, or 4 places. This shift is performed while the data is being transferred to the data bus for storage. The contents of the accumulator remain unchanged. When the high-order word is shifted left, the LSBs are transferred from the low-order word and the MSBs are lost

The accumulator also has the ability to simulate the effect of saturation in analog systems. This capability is implemented by using the accumulator overflow saturation mode, which is controlled by the OVM (overflow mode) status register bit. Setting the OVM bit with SOVM enables the accumulator saturation mode; resetting the OVM bit with ROVM disables it. If OVM is set and the accumulator operation results in an overflow, the accumulator is loaded with either the largest positive or negative number, depending on the sign of the operands and the actual result. The value of the accumulator upon saturation is 7FFFFFFFh (positive) or 80000000h (negative). If OVM is reset and an overflow occurs, the overflowed results are loaded into the accumulator without modification. (Note that logical operations cannot result in overflow.)

It is particularly desirable to enable the saturation mode when the accumulator contents represent a signal value. Without the saturation mode enabled, overflows cause undesirable discontinuities in the represented waveform. If the saturation mode is enabled, behavior of the accumulator when subjected to excessively large size signals resembles the tendency of an analog system to limit or saturate at a maximum level.

When an overflow occurs, the OV (overflow) bit in the status register is set, regardless of whether or not the OVM bit is set. The BV (branch on overflow) instruction, which branches only if OV is set, allows programs to make decisions (based on whether or not an overflow has occurred) and act accordingly. Once set, OV is reset only by the BV instruction or by directly loading the status register. Because OV is part of the status register, its state can be stored in data memory by using the SST (store status register) instruction or loaded by using the LST (load status register) instruction. This allows the state of OV from different program contexts to be saved independently, if desired, and examined outside of time-critical code segments.

The TMS320C1x can execute branch instructions that depend on the status of the ALU and accumulator. These instructions (BLZ, BLEZ, BGEZ, BGZ, BNZ, and BZ) cause a branch to be executed if a specific condition is met (see Chapter 4 for a complete list of TMS320C1x instructions).

3.4.3 Multiplier, T and P Registers

The TMS320C1x uses a 16 × 16-bit hardware multiplier (see Figure 3–18), which is capable of computing a 32-bit product in a single machine cycle. The following two registers are associated with the multiplier:

- A 16-bit temporary register (T) that holds one of the operands for the multiplier, **and**
- A 32-bit product register (P) that holds the product.

To use the multiplier, you must first load an operand into theT register from the data bus by using an LT, LTA, or LTD instruction. Then, use the MPY (multiply) or MPYK (multiply immediate) instruction to load the second operand (also from the data bus). If you use the MPY instruction, the multiplier value is a 16-bit number. If you use the MPYK instruction, the value is a 13-bit immediate constant contained in the MPYK instruction word. This13-bit constant is right-justified and sign-extended. After execution of the multiply instruction, the product is placed in the P register. The product can then be added to, subtracted from, or loaded into the accumulator by executing a PAC, APAC, SPAC, LTA, or LTD instruction. Pipelined multiply and accumulate operations can be performed with the LTA/LTD and MPY/MPYK instructions. Note that no provisions are made for the condition of 8000h × 8000h. If this condition arises, the product is 0C0000000h.

Note:

The contents of the P register cannot be restored without altering other registers. Interrupts are prevented from occurring until the instruction following the MPY/MPYK instruction has been executed. Therefore, the multiply instruction should always be followed by an instruction that combines the P register with the accumulator.

3.5 System Control

System control on the TMS320C1x processors is done by the program counter and stack, the external reset signal, interrupts (see Section 3.7), and the status register. This section explains the function of these components in system control. On the TMS320C17, a system control register controls the operation of the serial port, companding hardware, and the operation of the coprocessor port. The system control register for the TMS320C17 is discussed in Section 3.16. The SYSCON register for the TMS320C14 is shown in Figure 3–21.

Note:

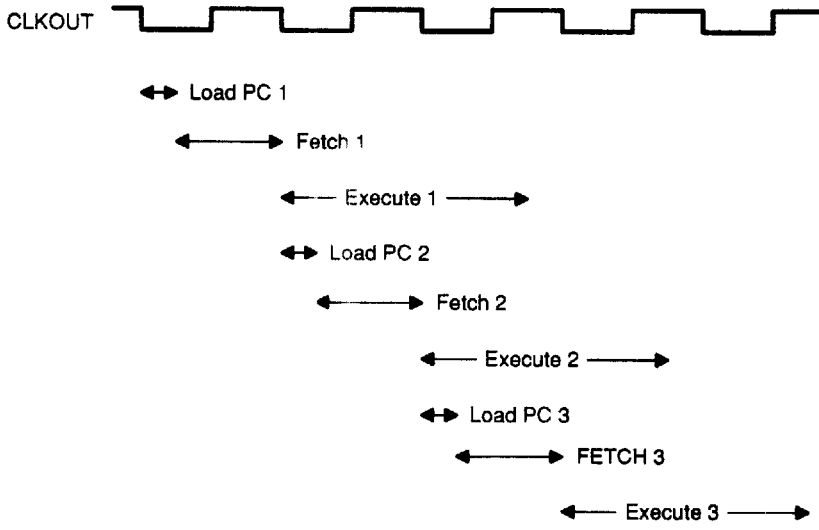
Where applicable, the program counter and address bus of the TMS320C16 has been increased to 16 bits. This allows a 64K program space which is larger than in other TMS320C1x devices. Also, the TMS320C17 is intended to only operate in microproccesor mode and does not have an external program address bus.

3.5.1 Program Counter and Stack

The program counter and stack enable the execution of branches, subroutine calls, interrupts, and table read/table write instructions. The program counter (PC) is a 12-bit register that contains the program memory address of the next instruction to be executed. The TMS320C1x reads the instruction from the program memory location addressed by the PC and increments the PC in preparation for the next instruction prefetch. The PC is initialized to zero by activating the reset ($\overline{RS}$) line.

The TMS320C1x devices utilize a modified Harvard architecture in which data memory and program memory lie in two separate spaces, thus permitting a full overlap of instruction fetch and execution. Figure 3–19 outlines the overlap of the instruction prefetch and execution. On the falling edge of CLKOUT, the program counter (PC) is loaded with the address of the instruction (load PC 2) to be prefetched while the current instruction (execute 1) is decoded and begins execution. The next instruction is then fetched (fetch 2) while the current instruction continues to execute (execute 1). Even as another prefetch occurs(fetch 3), both the current instruction (execute 2) and the previous instructions are still executing. This is possible because of a highly pipelined internal structure.

Figure 3–19. Instruction Pipeline Operation



To permit the use of external program memory, the PC outputs are buffered and sent to the external address bus pins A11 through A0. The PC outputs appear on the address bus during all modes of operation. The nine MSBs of the PC (A11 through A3) have unique outputs assigned to them, while the three LSBs are multiplexed with the port address lines, PA2 through PA0. The port address field is used by the I/O instructions, IN and OUT.

Program memory is always addressed by the contents of the PC. The contents of the PC can be changed by a branch instruction if the particular branch condition being tested is true. Otherwise, the branch instruction simply increments the PC. All branches are absolute, rather than relative: that is, a 12-bit value derived from the second word of the branch instruction is loaded directly into the PC to accomplish the branch. When interrupts or subroutine call instructions occur, the contents of the PC are pushed onto the stack to preserve return linkage to the previous program context. The second word also allows the TMS320C16 to branch to an absolute address within its 64K program space.

The stack is 12 bits wide and four levels deep. The PC stack is accessible through the use of the PUSH and POP instructions. The PUSH instruction pushes the twelve LSBs of the accumulator onto the top of the stack (TOS). Whenever the contents of the PC are pushed onto the TOS, the previous contents of each level are pushed down, and the fourth location of the stack is lost. Therefore, data is lost if more than four successive pushes (stack overflow) occur before a pop. The reverse happens on pop operations. The POP instruction

pops the TOS into the twelve LSBs of the accumulator. Any pop after three sequential pops yields the value at the fourth stack level. All four stack levels then contain the same value. Following the POP instruction, the TOS can be moved into data memory by storing the low-order accumulator word (SACL instruction). This allows expansion of the stack into data RAM. From data RAM, it can easily be copied into program RAM off-chip by using the TBLW (table write) instruction. In this way, the stack can be expanded to very large levels.

Note:

The TBLR and TBLW instructions utilize one level of the stack; therefore, only three nested subroutines or interrupts can be accommodated without overflowing the stack. For the TMS320C16 the stack has been increased to a depth of 8 and width of 16 to accomodate the larger program address space.

To handle subroutines and interrupts of much higher nesting levels, you can allocate part of the data RAM or external RAM to stack management. In this case, the TOS is popped immediately at the start of a subroutine or interrupt routine and stored in RAM. At the end of the subroutine or interrupt routine, the stack value stored in RAM is pushed back onto the TOS before returning to the main routine.

3.5.2 TMS320C1x Reset

Reset ($\overline{RS}$) is a nonmaskable external interrupt that can be used at any time to put the TMS320C1x into a known state. Reset is typically applied after powerup when the machine is in a random state. The reset input must be held low for a minimum of five clock cycles.

Driving the $\overline{RS}$ signal low causes the TMS320C1x to terminate execution and forces the program counter to zero. $\overline{RS}$ affects various registers and status bits. At powerup, the state of the processor is undefined. For correct system operation after powerup,a reset signal must be asserted low to guarantee a reset of the device (see Section 5.1 for other important reset considerations). Processor execution begins at location 0, which normally contains a B (branch) statement to direct program execution to the system initialization routine (see Section 5.1 for an initialization routine example).

When an $\overline{RS}$ signal is received, the following actions take place in the TMS320C1x (Except in the TMS320C14):

1) The control lines for $\overline{DEN}$, $\overline{WE}$, and $\overline{MEN}$ are forced high.
2) The data bus D15–D0 is placed in the high-impedance state.
3) The program counter (PC) is set to 0, and the address bus A11–A0 is driven with all zeros after the next clock cycle from $\overline{RS}$ going low.
4) The interrupt is disabled, and the interrupt flag register is reset to all zeros.

5) Control register bits on the TMS320C17 are set as follows: CR11 is set to 0, CR15 to 1, and CR29 to 0.

The TMS320C1x can be held in the reset state indefinitely. Note that the ARP, DP, and OVM status bits are not initialized by reset. Accordingly, it is critical that you initialize these bits in software, following reset.

3.5.3 TMS320C14 Reset

Upon receiving an $\overline{RS}$ signal, the following actions take place in the TMS320C14:

1) The program counter (PC) is set to 0, and the address bus A11–A0 is placed in the high-impedance state.
2) The control lines $\overline{WE}$ and $\overline{REN}$ are forced high.
3) The data bus D15–D0 is placed in the high-impedance state.
4) Bit I/O pins IOP15–IOP0 are configured as inputs and placed in the high-impedance state
5) Pins CMP4/CAP2 and CMP5/CAP3 are configured as capture inputs and placed in the high-impedance state.
6) Output pins CMP3–CMP0 are reset to 0.
7) The $\overline{WDT}$ pin is set to 1
8) Serial port pins TXD/CLK and RXD/DATA are placed in a high-impedance state.
9) The $\overline{NMI}$/MC/$\overline{MP}$ pin is sampled to determine whether internal or external program memory is enabled.
10) $\overline{RS}$ is brought high, and the address bus A11–A0 is cleared to all zeros in the next clock cycle.

In addition, Table 3–6 shows those registers that are also configured to the known state during reset. The status of all other registers is unknown at reset.

Table 3–6. Registers Configuration on Reset (TMS320C14)

Register	Port	Description
BSR	FFFFh	Bank select register. Points to off-chip peripherals
DDR	0000h	Data direction. All bit I/O pins configured as inputs.
IF	0000h	Interrupt flag register. All interrupt flags cleared
IM	FFFFh	Interrupt mask register. All interrupts masked or disabled.
WDT	0000h	Watchdog timer. Set to 0.
WPER	FFFFh	Watchdog period. Set to maximum count.
WTPL	FFFFh	Watchdog timer period latch. Set to maximum count.
TCON	0000h	Timer control register. Timers 1 and 2 disabled. Compare pins held at 0. Compare disabled. Capture system enabled.
CCON	0000h	Capture control register. Capture on all pins disabled.

3.5.4 Status Register

The status register consists of five status bits. These status bits can be individually altered through dedicated instructions. In addition, the SST instruction stores the status register in data memory. The LST instruction loads the status register from data memory, with the exception of the INTM bit. This bit can be changed only by the EINT/DINT (enable/disable interrupt) instructions. In this manner, the current status of the device may be saved on interrupts and subroutine calls.

Table 3–7 shows instructions that affect the status register contents. Note that several bits in the status registers are reserved and read from the status register as logic ones by the SST instruction.

Table 3–7. Status Register Field Definitions

Field	Function
ARP	Auxiliary Register Pointer. This single-bit field selects the AR to be used in indirect addressing. ARP =0 selects AR0; ARP = 1 selects AR1. ARP may be modified by executing instructions that permit the indirect addressing option, and by executing the LARP, MAR, and LST instructions.
DP	Data Memory Page Pointer. The single-bit DP register is concatenated with the 7 LSBs of an instruction word to form a direct memory address of 8 bits. DP = 0 selects the first 128 words of data memory: that is, page 0. DP = 1 selects page1, the remaining words in data memory. DP may be modified by the LST, LDP, and LDPK instructions.
INTM	Interrupt Mode Bit. When an interrupt is serviced, the INTM bit is automatically set to one before the interrupt service routine begins. INTM = 0 enables all maskable interrupts; INTM = 1 disables all maskable interrupts. INTM is set and reset by the DINT and EINT instructions, respectively. $\overline{RS}$ also sets INTM. INTM has no effect on the unmaskable $\overline{RS}$ or $\overline{NMI}$ interrupts. Note that INTM is unaffected by the LST instruction.
OV	Overflow Flag. OV = 0 indicates that the accumulator has not overflowed. OV = 1 indicates that an overflow has occurred. Once an overflow occurs, the OV remains set until a reset, BV, or LST instruction clears the OV.
OVM	Overflow Mode Bit OVM = 0 disables the overflow mode, causing the overflowed results to remain in the accumulator. OVM = 1 enables the overflow mode, causing the accumulator to be set to either its most positive or most negative value upon encountering an overflow. The SOVM and ROVM instructions set and reset this bit. LST may also be used to modify the OVM.

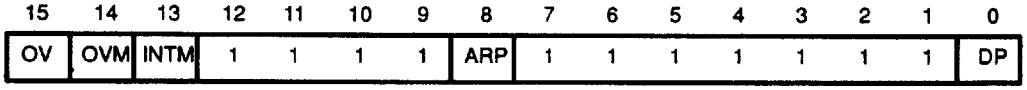
The contents of the status register can be stored in data memory by executing the SST instruction. Thus, an SST instruction using the direct addressing mode can specify only an address less than 16 on the TMS320C10 because the second page of memory contains only 16 words. The second page of memory on the TMS320C15 and TMS320C17 contains 128 words. If the indirect addressing mode is selected, the contents of the status register may be stored in any RAM location selected by the auxiliary register.

Note:

If the SST instruction is executed using the direct addressing mode, the device automatically stores this information on page 1 of data memory at the location specified by the instruction, regardless of the value of the data page pointer.

The SST instruction does not modify the contents of the status register. Figure 3–20 shows the position of the status bits as they appear in the appropriate data RAM location after execution of the SST instruction.

Figure 3–20. Status Register Organization



The LST instruction maybe executed to load the status register. LST does not assume status bits are on page one. When direct memory addressing has been used, the DP must be set to one for the LST instruction to access status bits stored on page one.

Note:

The LST instruction cannot change the interrupt mode (INTM) bit; however, it can change all other status bits.

3.5.5 System Control on the TMS320C14

System control on the TMS320C14 processors is provided by the program counter and stack, the SYSCON register (mode control), the external reset signal, the status register, and the interrupts. The functionality of the program counter, stack, and status registers is common to all TMS320C1x devices and is explained in Section 3.5. This section explains the functions of the mode control, reset, and interrupt components that are specific to the TMS320C14.

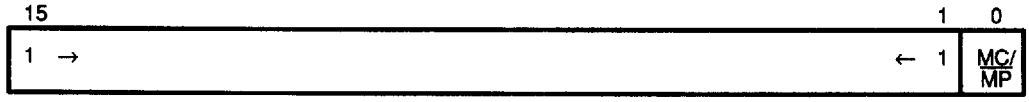
Note:

The TMS320C14 does not have the BIO pin that is present on other TMS320C1x devices. An attempt to execute the BIOZ (Branch on BIO low) instruction results in a two-cycle NOP action.

3.5.5.1 Microprocessor/Microcomputer Modes and Reset

The TMS320C14 has two modes of operation: microprocessor and microcomputer. These modes are controlled by the MC/$\overline{\text{MP}}$ (bit 0) of the SYSCON register (see Figure 3–21). Bits 15–1 in this register must be set to all ones.

Figure 3–21. SYSCON Register



Writing a 1 to the MC/$\overline{\text{MP}}$ bit configures the device to microcomputer mode. In this mode, all program memory accesses are performed from internal 4K-

word program memory. The 16-bit external data bus can be used for accessing off-chip peripherals. However, only the lower three address lines (PA0–PA2) are driven, while the upper 9 address lines are always 1. Writing a 0 to the MC/$\overline{MP}$ bit (bit 0) of the SYSCON register configures the device into microprocessor mode. In this mode, all program accesses are made from external memory, and the internal ROM/EPROM is disabled.

In addition to the software control of the MC/$\overline{MP}$ bit in the SYSCON register, the TMS320C14 has an external hardware option that controls this bit and configures the device in microprocessor or microcomputer mode. The $\overline{NMI}$/MC/$\overline{MP}$ pin is sensed while the $\overline{RS}$ pin is low. If the $\overline{NMI}$/MC/$\overline{MP}$ pin is low at that time, internal program memory is disabled and all program accesses are from external memory. If the $\overline{NMI}$/MC/$\overline{MP}$ pin is high at the time, the device is placed in the microcomputer mode and all program memory accesses are from internal memory. Once $\overline{RS}$ goes high and this pin is brought high, it behaves as a normal NMI pin. The $\overline{NMI}$/MC/$\overline{MP}$ (nonmaskable interrupt) is edge triggered, and the pin can be brought high anytime after a reset without generating an interrupt. For more configuration information, see subsection 6.5.3 on page 6-2.

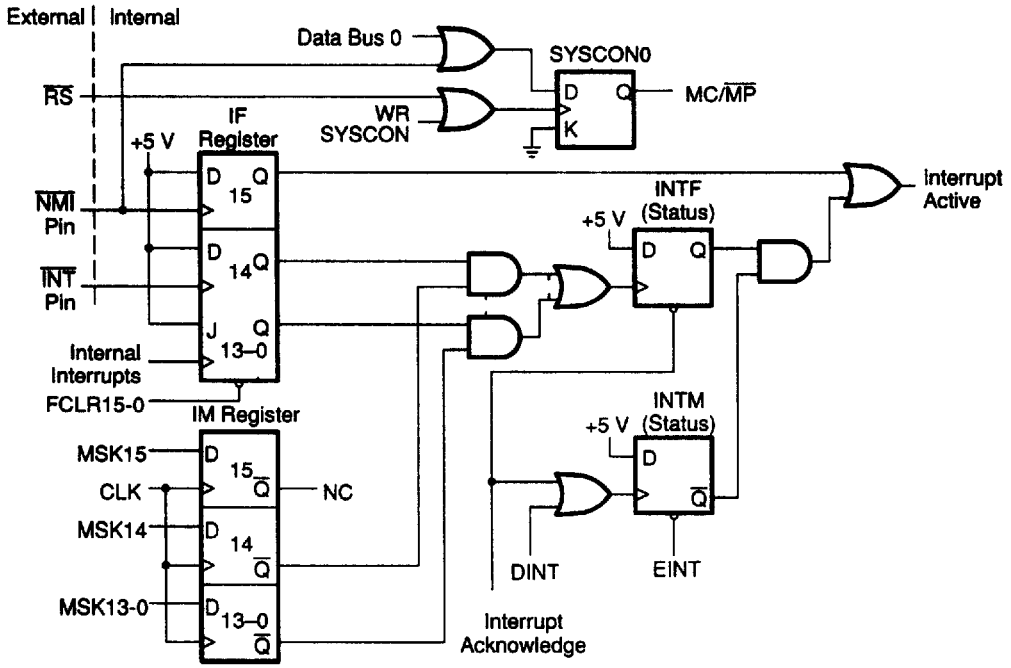
Even if you initially configure the TMS320C14 hardware to be in microcomputer or microprocessor mode, you can still modify the MC/$\overline{MP}$ bit to change the mode on the device. Using a software bit, in addition to the hardware option, provides much greater flexibility. Changing the microprocessor/microcomputer modes with software allows you to switch from internal to external memory, thus doubling the memory size to 8K words.

To provide an orderly switch of program memory, a delay of two cycles is needed. The context switches two cycles after writing to the MC/$\overline{MP}$ bit (using an OUT instruction). The two-cycle delay allows a B (branch), CALL, or RET instruction to be executed and reach the desired memory location. If no location change is desired, two NOPs should be introduced to track the two-cycle delay. The SYSCON register can be accessed at bank 1h and port address 3h.

3.5.6 TMS320C14 Interrupts

The TMS320C14 provides a total of 15 external and internal interrupts for communication with time-critical internal and external operations. Two interrupts are dedicated for external sources and are triggered by a negative edge on pins $\overline{NMI}$/MC/$\overline{MP}$ and $\overline{INT}$. The remainder of the interrupts are used to service the on-chip peripherals. All the interrupts, internal or external, are mapped into a 16-bit register called the interrupt flag register (IF). In addition, a 16-bit register called the interrupt mask register (IM) is also available to mask individual interrupts. Figure 3–22 shows the architecture of the interrupt subsystem.

Figure 3–22. TMS320C14 Interrupt Subsystem



When any interrupt (except NMI) is generated by either a peripheral or an external source, the following sequence of events occurs:

1) A bit is set to 1 in the IF register corresponding to that interrupt. This indicates an interrupt pending.
2) A check is made to determine if the corresponding bit in the IM register is 0. This indicates an interrupt is unmasked.
3) If the interrupt is unmasked, an interrupt is generated to the CPU by setting the interrupt flag (INTF). If the interrupt is masked, it continues to be latched in the IF register, and interrupts the CPU only when unmasked.
4) If the interrupt mode (INTM) bit in the status register is 0 (CPU interrupt is enabled), the CPU responds by saving the present program counter value (PC) on the hardware stack and branching to location 2 in program memory. If the INTM bit in the status register is 1, the CPU interrupt continues to be latched in the INTF bit.
5) Sequence of events 1 through 4 is true for all interrupts except the non-maskable interrupt (NMI). In the case of the NMI, the interrupt is passed straight through to the CPU without checking the IM register or the INTM bit. The CPU responds immediately by saving the present PC value on the hardware stack and branching to location 2 in program memory.
6) The INTM bit is set to 1, disabling further interrupts (except NMI).

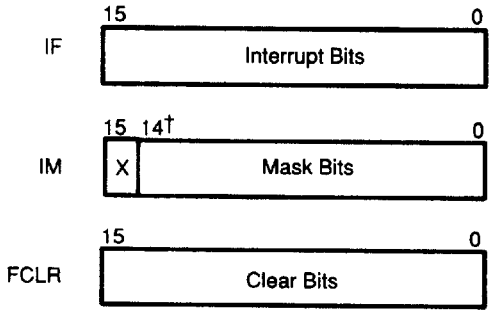
7) The interrupt service routine, starting at address 2h, polls the IF flag register to determine which peripheral is the source of the interrupt. In addition, the corresponding flag in the IF flag register must be cleared and the CPU interrupt enabled by executing an enable interrupt (EINT) instruction.

To facilitate clearing individual flags in the IF register, an additional register called the flag clear, (FCLR) is also provided. When a 1 is written to a bit in the FCLR register, it clears the corresponding bit in the IF register. Writing a 0 to a bit in the FCLR register leaves the corresponding bit in the IF register unaffected. The IF, IM, and FCLR registers have a bank address of 0h and port addresses of 4h, 5h, and 6h, respectively. Figure 3–23 shows the relationship of the IF, IM and FCLR registers.

Note:

The IF register should not be written to directly. To reduce the risk of affecting the wrong bits, all writes should be through the FCLR register.

Figure 3–23. IF/IM/FCLR Register Relationship



† Most significant *usable* bit. Writing a 1 to IM bit 15 does not mask corresponding interrupt (NMI)

Figure 3–24 and Table 3–8 describe the individual interrupt bits of the IF register. Detailed descriptions of how these interrupts are generated is found in the section describing peripheral operation. When an interrupt is received, the corresponding bit in the IF register is set to 1. This IF register bit remains set until cleared. To mask an interrupt, the corresponding bit in the IM register is set to 1. The mask remains in effect until the IM bit is cleared.

Figure 3–24. IF Register

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
NMI	INT	†	TIMINT3/ STMRINT	CAP-INT3	CAP-INT2	CAP-INT1	CAP-INT0	CMP-INT1	CMP-INT0	TIM-INT2	TIM-INT1	RXINT	TXINT	WDT INT	IOPINT

† Reserved bit

Table 3–8. IF Register Description

Bit	Interrupt	Description
15	NMI	Nonmaskable interrupt
14	INT	External interrupt
13	Reserved	IF 13 should be set to 0, IM 13, to 1
12	TIMINT3/STMRINT	Timer 3 interrupt (Timer 3 is normally used by serial port)
11	CAPINT3	Capture interrupt 3
10	CAPINT2	Capture interrupt 2
9	CAPINT1	Capture interrupt 1
8	CAPINT0	Capture interrupt 0
7	CMPINT1	Compare interrupt 1
6	CMPINT0	Compare interrupt 0
5	TIMINT2	Timer 2 interrupt
4	TIMINT1	Timer 1 interrupt
3	RXINT	Serial port receive interrupt
2	TXINT	Serial port transmit interrupt
1	WDTINT	Watchdog timer interrupt
0	IOPINT	I/O port IOP interrupt

3.6 Input/Output Functions TMS320C1x

The TMS320C1x implements a variety of different I/O functions to communicate with external devices. The 16-bit parallel data bus can perform I/O functions in two cycles with the IN and OUT instructions. The I/O ports are addressed by the three LSBs of the address bus (PA2–PA0). In addition to I/O functions, a polling input ($\overline{BIO}$) for both bit test and branch operations and an interrupt input ($\overline{INT}$) have been incorporated for increased system flexibility.

I/O design is simplified by treating I/O the same way as memory. I/O devices are mapped into the I/O address space by using the processor's external address and data buses in the same manner as memory-mapped devices.

The TBLR and TBLW instructions transfer words between program and data spaces. TBLR is used to read words from on-chip ROM or off-chip program ROM/RAM into the data RAM. TBLW is used to write words from on-chip data RAM to off-chip program RAM on the TMS320C10 and the TMS320C15. External program memory cannot be addressed on the TMS320C17.

3.6.1 Input/Output Functions (TMS320C10/C15)

Executing IN and OUT instructions inputs/outputs data to and from a peripheral. Data is transferred over the 16-bit data bus to and from data memory by two independent strobes: data enable ($\overline{DEN}$) and write enable ($\overline{WE}$).

The bidirectional external data bus is always in the high-impedance state, except when $\overline{WE}$ is active (low), or during an IN instruction from port 0 or port 1 on the TMS320C17 (see subsection 3.6.4). $\overline{WE}$ goes low during the first cycle of the OUT instruction and the second cycle of the TBLW instruction.

The three port address pins (PA2–PA0) output the port address during IN and OUT instructions. Execution of an IN instruction generates the $\overline{DEN}$ strobe for transferring data from a peripheral device to the data RAM (see Figure 3–26). The IN instruction is the only instruction for which $\overline{DEN}$ becomes active. Execution of an OUT instruction generates the $\overline{WE}$ strobe for transferring data from the data RAM to a peripheral device (see Figure 3–27). $\overline{WE}$ becomes active only during the OUT and TBLW (table write) instructions; see Appendix A for timing information.

When the three multiplexed LSBs of the address bus (PA2–PA0) are used as a port address by the IN or OUT instruction, the remaining higher-order bits of the address bus (A11–A3) are held at logic zero during execution of either instruction.

Eight I/O addresses are available on the TMS320C10 and TMS320C15 for interfacing to peripheral devices: eight 16-bit multiplexed input ports and eight 16-bit multiplexed output ports (see Figure 3–25).

Figure 3–25. TMS320C1x External Device Interface

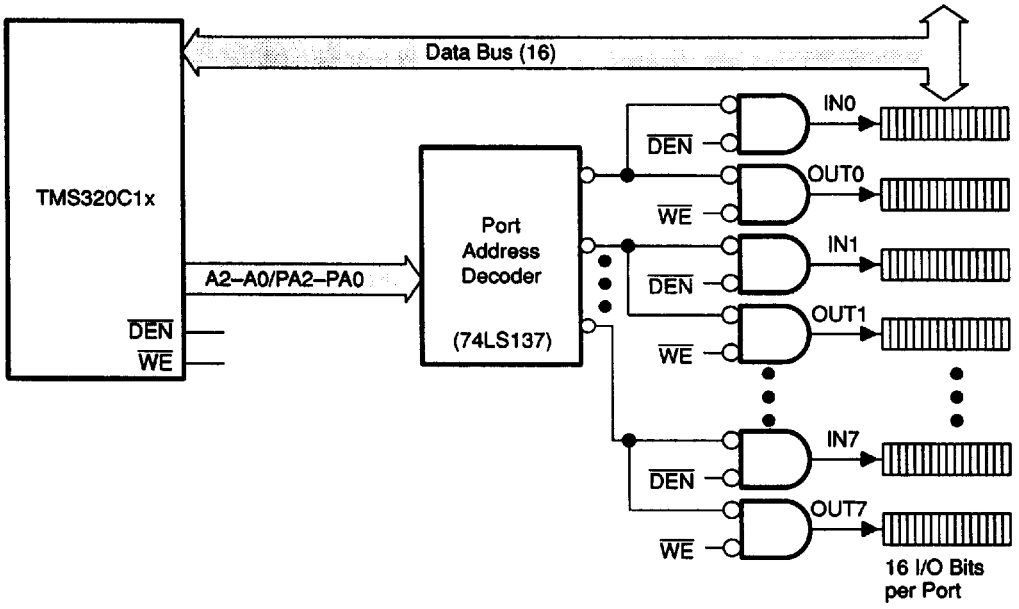


Figure 3–26. Input Instruction Timing

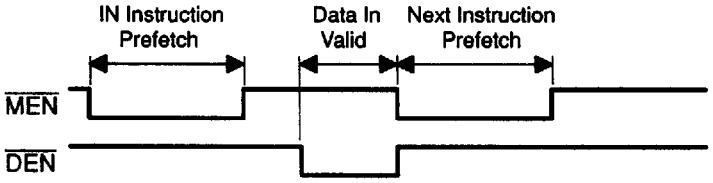
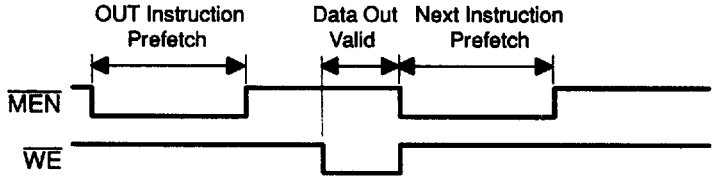


Figure 3–27. Output Instruction Timing



Execution of the TBLR instruction generates $\overline{\text{MEN}}$ strobes to read the word from program memory (see Figure 3–28). Execution of a TBLW instruction generates a $\overline{\text{WE}}$ strobe (see Figure 3–29). Note that the data bus is driven and the $\overline{\text{WE}}$ strobe is generated even if the device is in the microcomputer mode and a TBLW is performed to a program location residing in on-chip ROM.

Figure 3–28 and Figure 3–29 show a prefetch of the instruction following the TBLR or TBLW instruction. The first prefetch is discarded and a second prefetch occurs at the end of the TBLR or TBLW instruction. The $\overline{MEN}$, $\overline{DEN}$, and $\overline{WE}$ interface strobes are mutually exclusive.

Figure 3–28. TBLR Instruction Timing

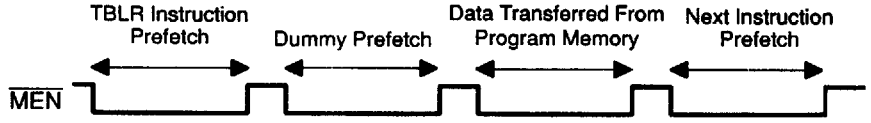
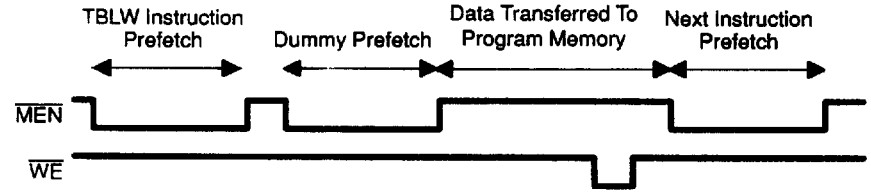


Figure 3–29. TBLW Instruction Timing



Note:

The following are important considerations for those designs that use program memory. Because the OUT and TBLW instructions use only the $\overline{WE}$ signal to indicate valid data, these instructions cannot be distinguished from one another on the basis of the interface strobes. Execution of a TBLW instruction writes data to peripherals, and execution of the OUT instruction overwrites program memory. Because it is impossible to use TBLW to uniquely write to program memory locations, it is advisable to avoid mapping both I/O and external program RAM into the same locations.

The prgram memory locations that are in conflict are as follows:

Device	pma	
'C10/'C15/'C16	0–7	(0h–7h)
'C14	4088–4095	(FF8h–FFFh)
'C17	Not Applicable (No external program space)	

3.6.2 General-Purpose $\overline{BIO}$ Input Pin (TMS320C10/C15/C16)

The $\overline{BIO}$ pin is an external general-purpose software-controlled input pin that supports bit test and branch operations.

When the $\overline{BIO}$ input pin is active (low), execution of the BIOZ instruction causes a branch to occur. The $\overline{BIO}$ pin is useful for monitoring peripheral device status. It is especially advantageous as an alternative to an interrupt when time-critical loops must not be disturbed.

For systems using asynchronous inputs to the $\overline{BIO}$ pin on a TMS32010 (obsolete NMOS device), external hardware is required to ensure proper execution of the BIOZ instruction. This hardware synchronizes the $\overline{BIO}$ input signal with the rising edge of CLKOUT on the TMS32010. See Appendix A for information regarding this system design consideration.

3.6.3 General-Purpose XF Output Pin (TMS320C17)

The XF (external flag) output pin on the TMS320C17 is an external logic output flag. Programmed through control register bit 10 (CR10), this pin is the direct output of the CR10 latch. When the CR10 bit is set to a 1, the XF pin is set to a logic high; when CR10 is reset to a 0, the XF pin is driven low.

3.6.4 Input/Output Functions (TMS320C17)

Because the system control register, serial port transmit and receive registers, and companding hardware have been mapped into I/O ports 0 and 1, only six input and six output ports are available on the TMS320C17 for interfacing to peripheral devices.

On the TMS320C17, the purpose and usage of pins PA2 through PA0 depends on the selected mode of operation: microcomputer or microprocessor. (MC/$\overline{MP}$). In the microcomputer mode, pins PA2–PA0 output the three LSBs of the program counter, except during IN and OUT instructions. During IN and OUT instructions, these three pins address the serial port, companding hardware, and off-chip I/O peripherals. During reset, the pins along with the program counter are synchronously cleared to zero during the cycle following $\overline{RS}$ low. Because program and data memories are contained on-chip, only these three address lines are output from the device.

The memory enable ($\overline{MEN}$) signal is not implemented on the TMS320C17 devices, because all instruction execution is from on-chip program ROM. Additionally, the bidirectional external data bus on theTMS320C17 is always in the high-impedance state, except when $\overline{WE}$ is active (low) or during an IN instruction from port 0 or port 1. $\overline{WE}$ goes low during the first cycle of the OUT instruction to provide the write strobe for writing data to a peripheral.

In the coprocessor mode, pins PA2 –PA0 have different functions, respectively referred to as $\overline{TBLF}$ (transfer buffer latch full), $\overline{RBLE}$ (receive buffer latch empty), and HI/$\overline{LO}$ (high or low transfer select of an 8-bit byte). Except for IN and OUT instructions to Port 5, no other activity occurs on these pins. In this mode, the IN and OUT instructions to Ports 0 and 1 also provide the processor with an access to the on-chip serial ports and companding hardware and to the coprocessor port latches. The data bus is in a high-impedance state, unless $\overline{RD}$ is active (low).

On the TMS320C17, the system control register (see Section 3.16), serial port receive and transmit registers (subsections 3.13.1 and 3.13.2), and the com-

panding hardware (Section 3.14) have been mapped into I/O ports 0 and 1. During an OUT or IN instruction to port 0 or port 1, data appears on the external data bus (D15–D0). The data bus is not in the high-impedance state while accessing these dedicated I/O ports. Peripheral device interface should be to port addresses 2 through 7 to prevent bus conflicts with the system control register and serial port. Six 16-bit multiplexed input ports and six 16-bit multiplexed output ports are available for interfacing to peripheral devices.

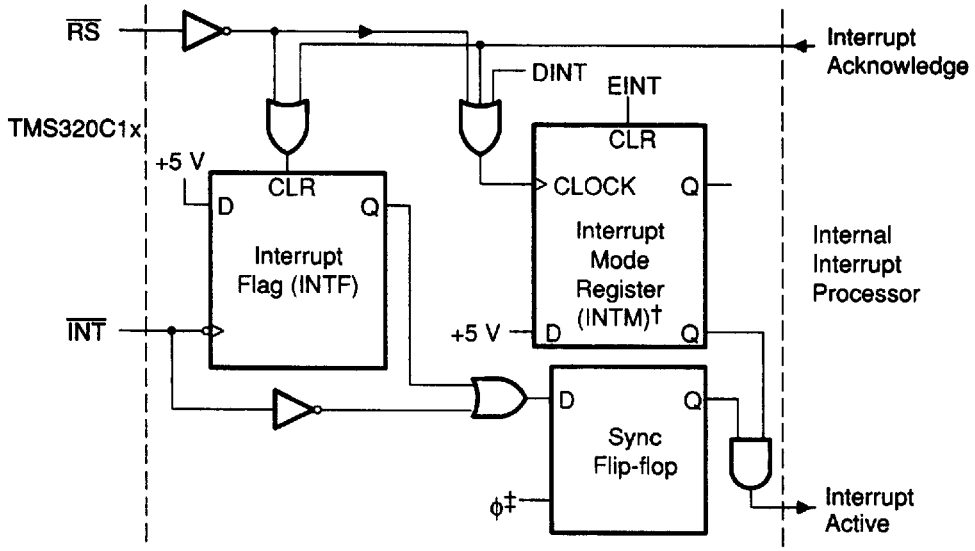
The XF (external flag) output pin, specific to the TMS320C17, is an external logic flag to implement single-bit digital output. Programmed through control register bit 10 (CR10), this pin is the direct output of the CR10 latch. When the CR10 bit is set to a 1, the XF pin is set to a logic high; when CR10 is reset to a 0, the XF pin is driven low.

3.7 Interrupts

This section describes the interrupt for the TMS320C1x generation. The TMS320C1x provides an external interrupt input for communication with time-critical external operations. The interrupt can be generated by applying either a negative-going edge or a logic low level to the interrupt input pin. All interrupts on the TMS320C1x are maskable through the use of the status register interrupt mode bit and various mask bits.

A simplified diagram of the internal interrupt circuitry for TMS320C1x CMOS devices is shown in Figure 3–30. Note that the TMS32010 requires external synchronizing flip-flops on interrupts and $\overline{BIO}$. These synchronizing flip-flops are not required on the TMS320C10/C15/C17.

Figure 3–30. TMS320C1x Simplified Interrupt Logic Diagram



† Q = 0 indicates interrupts enabled. Q = 1 indicates interrupts disabled.

‡ ϕ = phase of internal clock.

Note: The TMS32010 requires external synchronizing flip-flops.

Interrupt servicing begins with the following sequence of events:

1) The interrupt is acknowledged, which clears the INTF (interrupt flag) bit to 0.
2) The INTM (interrupt mode) bit is set to 1 to disable further interrupts.
3) The current PC is pushed onto the TOS (top of stack).
4) The new PC is set to 2.

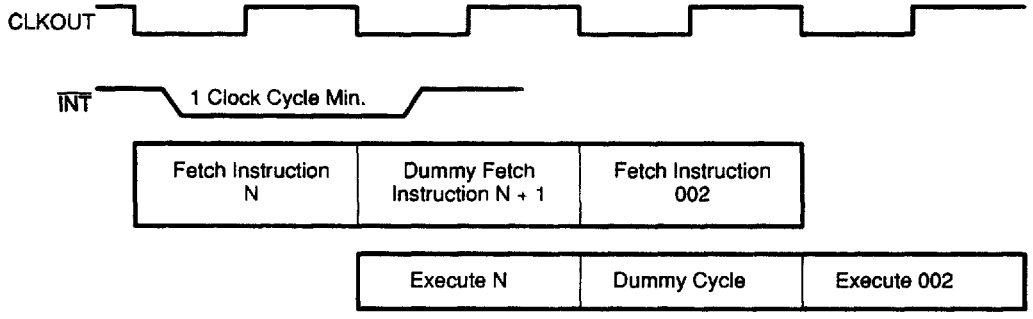
You begin interrupt servicing at program memory address 2h. At the end of the interrupt servicing, you execute an EINT instruction to clear the INTM register (set to zero) to enable the interrupts. A DINT instruction or a hardware reset also sets the INTM register to one (see Figure 3–30), disabling interrupts. You must execute an EINT instruction to enable interrupts again.

Note that interrupt servicing is delayed until:

1) The end of all cycles of a multicycle instruction,
2) The instruction following the MPY or MPYK instruction has completed, or
3) The instruction following the EINT instruction has been executed (when interrupts have been previously disabled). This allows the RET instruction to be executed after interrupts become enabled at the end of an interrupt routine.

Figure 3–31 shows the instruction sequence that occurs, once an interrupt becomes active. The dummy fetch is an instruction that is fetched but not executed. This instruction is refetched and executed after the interrupt routine is completed.

Figure 3–31. Interrupt Timing



Two steps must be taken to enable an active interrupt to the device. First, the individual interrupt must be enabled by writing a logic 1 to the appropriate system control register bit (CR7–CR4). Then, the master interrupt circuitry is enabled via the EINT instruction. An interrupt flag represents a valid interrupt con-

dition to the processor if interrupts have been enabled. Thus, before enabling interrupts, the flag bits of all undesired interrupts should be cleared.

3.7.1 Interrupts (TMS320C17)

The TMS320C17, in addition to the external interrupt for communication with time-critical external operations, also has three additional internal interrupts, which are generated by the two serial ports. When the TMS320C17 is operating in the coprocessor mode, the $\overline{\text{EXINT}}$ and $\overline{\text{BIO}}$ pins are ignored. Internally, comparable signals are supplied as a result of pulses on the $\overline{\text{RD}}$ and WR pins in the coprocessor mode.

The TMS320C17 has four maskable interrupts: $\overline{\text{EXINT}}$, $\overline{\text{FSR}}$, $\overline{\text{FSX}}$, and FR. On this device, the interrupt function of the TMS320C10 and TMS320C15 has been expanded to support the serial port interface. An interrupt latch and multiplexer is used to generate the master interrupt signal, which functions identically to the $\overline{\text{INT}}$ interrupt on the TMS32010. Thus, all the maskable interrupts have the same priority and require the use of interrupt polling techniques when multiple interrupts are enabled.

When interrupts are enabled, an interrupt becomes active either because of a low-voltage input on the $\overline{\text{INT}}$ pin or when a negative edge has been latched into the interrupt flag (INTF). If the interrupt mode register (INTM) is set to zero, an interrupt active signal to the internal interrupt processor becomes valid.

In the coprocessor port mode, the external interrupt ($\overline{\text{EXINT}}$) flag cannot be cleared until four cycles after the data from the coprocessor port has been read. In a reset initialization routine, the interrupt flag bits (CR3–CR0) should be cleared before the EINT instruction to insure that a false interrupt does not occur (see Section 3.12 for detailed interrupt bit descriptions).

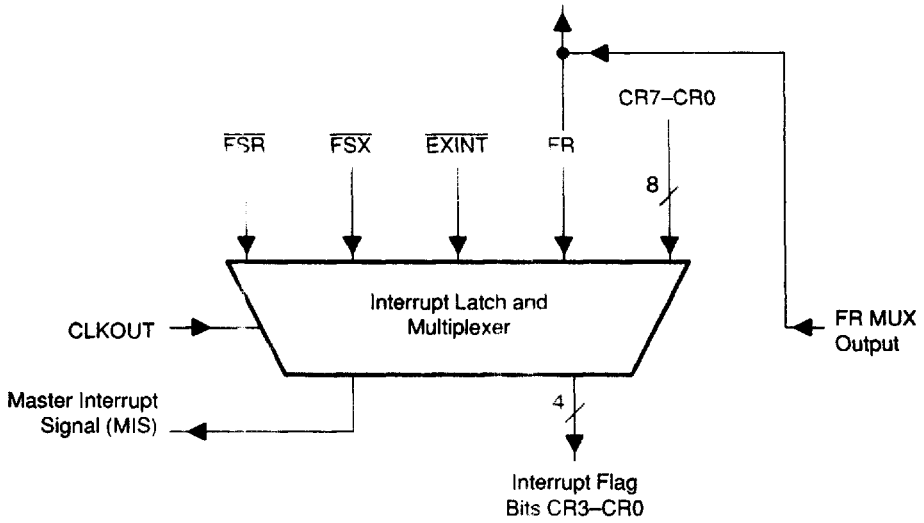
The interrupt latch synchronizes all interrupts to the device output clock (CLKOUT). A block diagram of the interrupt latch and multiplexer is shown in Figure 3–32.

The external interrupt flag ($\overline{\text{EXINT}}$) is set by one of two conditions: (1) an asynchronous input to the device for external control, or (2) a master processor interrupt signal when the TMS320C17 is being operated in coprocessor mode. The $\overline{\text{EXINT}}$ flag cannot be cleared while an interrupt condition is presented.

The other three interrupts are normally associated with the serial port framing signals. A bit in the system control register (CR9) designates whether FR or, alternatively, $\overline{\text{FSX}}$ and $\overline{\text{FSR}}$ are to be used for framing. When $\overline{\text{FSX}}$ and $\overline{\text{FSR}}$ control the serial port framing, FR can function as an independent timer interrupt with the timer clocked by the SCLK source. When the serial port is controlled by the internal framing pulse (FR), the $\overline{\text{FSX}}$ and $\overline{\text{FSR}}$ inputs are available as independent edge-triggered interrupts.

Due to the asynchronous operation of the interrupts, the time between the occurrence of an active interrupt signal and the device actually vectoring to ROM location 2 is four CLKOUT cycles; see Appendix A for further timing information

Figure 3–32. Interrupt Latch and Multiplexer



3.8 Peripherals (TMS320C14)

The TMS320C14 includes all the features of the TMS320C1x achitecture as well as some additional I/O functions. The 16-bit parallel data bus can be utilized to access external program memory and I/O functions. These external bus cycles are controlled by the write enable ($\overline{WE}$) and read enable ($\overline{REN}$) pins.

The TMS320C14 has 16 pins of bit I/O that can be individually selected as inputs or outputs. There are provisions to allow setting and clearing of each pin without affecting the others. The capability to detect and match patterns on the input pins is also included. Refer to Section 3.9 for more information on the bit I/O pins.

Also included in the TMS320C14 are two 16-bit timers that can be used as event counters with internal or external clocks, and a watchdog timer that is available for time-out functions. A fourth timer, the serial port baud rate generator, is intended for serial port operation, but may also be used as a general-purpose timer if serial port communication is not used. Associated with each timer is a 16-bit period register. Refer to Section 3.6 for more information on the timers.

The TMS320C14 has an event manager that consists of a compare subsystem and a capture subsystem. The compare subsystem has six compare registers that constantly compare their outputs with one of the timers. Associated with each compare register is an action register that controls the compare output pins. The action registers determine actions that take place on output pins in case of a match between the timer and a compare register. In addition, the compare subsystem can be configured to generate six channels of high-precision PWM. The event manager also contains four capture inputs. This subsystem captures the value of a timer in a corresponding four-deep FIFO stack when a certain transition is detected on a capture input pin. Section 3.11 contains more information on the event manager.

The serial port of the TMS320C14 provides an asynchronous mode of communication with other devices. Refer to Section 3.12 for more information on the serial port.

The TMS320C14 has a total of 15 internal/external interrupts that can be individually masked. An external nonmaskable interrupt ($\overline{NMI}$/MC/$\overline{MP}$) is also available. Each of the interrupts triggers a master interrupt. The master interrupt is controlled by the INTM bit in the status register. For more information, refer to subsection 3.5.6 regarding interrupts.

A maximum of eight I/O addresses are available on the TMS320C1x and TMS320C14 for interfacing to peripheral devices: eight 16-bit multiplexed input ports and eight 16-bit multiplexed output ports. To maintain compatibility with the TMS320C1x architecture, all peripherals on the TMS320C14 are implemented in the I/O address space. However, the number of on-chip

peripheral devices that can be accessed is greater than 16. To implement the peripherals within the address space of eight I/O ports, a dual address scheme has been adopted that includes a bank address, and a port address within that bank. This allows you to preserve your investment in TMS320C1x software and development tools.

3.8.1 On-Chip Peripheral Register Mapping

The on-chip peripherals on the TMS320C14 are controlled by peripheral registers mapped in the I/O space. A 16-bit bank select register (BSR) is used to map all these internal peripherals in the I/O space. Each peripheral register has a bank address and a port address. The bank address is selected by writing the bank address into the bank select register (BSR). The port address is specified in the IN or OUT instruction.

Table 3–9 shows the location of each peripheral register.

Table 3–9. I/O Register Map

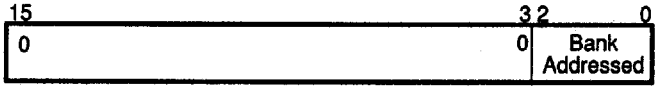
Port	Bank0	Bank1	Bank2	Bank3	Bank4	Bank5	Bank6	Bank7	BankFFFF
0	IOP	WDT	TMR1	CMPR0	ACT0	SCON	FIFO0		Ext. I/O
1	DDR	WPER	TPR1	CMPR1	ACT1	SSET	FIFO1		Ext. I/O
2	BSET	WTPL4	TMR2	CMPR2	ACT2	SCLR	FIFO2	SMAT	Ext. I/O
3	BCLR	SYSCON	TPR2	CMPR3	ACT3	TBR	FIFO3	TSR	Ext. I/O
4	IF		TCON	CMPR4	ACT4	RBR	CCON	RSR	Ext. I/O
5	IM			CMPR5	ACT5	SBRG	CCLR	STMR	Ext. I/O
6	FCLR								Ext. I/O
7	BSR	BSR	BSR	BSR	BSR	BSR	BSR	BSR	BSR

3.8.2 On-Chip/Off-Chip Peripheral Selection

To select a particular register, first store the bank address in the BSR, enabling the selected bank. Then an access to that port is made with an IN or OUT instruction. If another register in the same bank needs to be accessed, it is not necessary to write to the bank register again. For an OUT instruction, data always appears on the data bus. The $\overline{WE}$ strobe, however, is activated only when an external I/O (EXTADDRESS = FF8h = FFFh).

Port 7 is reserved in all banks for the BSR. Any I/O read/write to port 7 using IN or OUT instructions automatically accesses the bank select register (BSR). As shown in Figure 3–33, only the lower three bits of the BSR are used to select the bank address for the on-chip peripherals. The upper 13 bits must be zeros.

Figure 3–33. Bank Select Register



To select off-chip peripherals, you must use the bank address FFFFh. Table 3–10 lists the peripheral registers available on the TMS320C14 and their port and bank addresses. More information about the operation of the peripheral registers is given in the detailed descriptions of the on-chip peripherals.

Note:

External I/O access uses the same control lines used in TBLW and in program fetch. This aliases the I/O space with the program space. Be careful to avoid conflict.

Table 3–10. Peripheral Registers

Register	Port	Bank	Description
SYSCON	3	1	System configuration register. Configures the device as microcomputer or microprocessor.
BSR	7	All	Bank select register. Stores bank address for a register. All read/writes for port 7 go automatically to BSR. Used in selection of on-chip/off-chip peripherals.
Bit I/O Ports			
IOP	0	0	I/O port latch. Stores data for IOP pins configured as output. Also stores data for pattern match on input IOP pins.
DDR	1	0	Data direction register. Configures IOP pins as inputs or outputs.
BSET	2	0	Bit set register. Allows setting of individual bits in IOP latch without affecting other bits.
BCLR	3	0	Bit clear register. Allows clearing of individual bits in IOP latch without affecting other bits.
Interrupts			
IF	4	0	Interrupt flag register. Indicates interrupts that have been received by the device.
IM	5	0	Interrupt mask register. Directs the CPU to acknowledge or ignore an interrupt source.
FCLR	6	0	Flag register clear. Allows individual clearing of bits in IF register without affecting other bits.
Timers			
WDT/ TMR4	0	1	Watchdog timer/Timer 4. Free-running 16-bit timer that acts as a watchdog timer. Can also be used as a fourth timer. Timer is reset to 0000h when it equals the period register.
WPER/ TPR4	1	1	Watchdog timer period register. Causes the timer to reset to 0000h when its value equals the period

Table 3–10. Peripheral Registers(Continued)

Register	Port	Bank	Description
WTPL	2	1	Watchdog timer period register latch that prevents the watchdog period from being changed on the fly. This register stores the old value of WPER, used by WDT to determine its period.
TMR1	0	2	Timer 1. 16-bit timer that can be used as an event counter. TMR1 is reset to 0000h when its value equals TPR1.
TPR1	1	2	Timer 1 period register. Causes TMR1 to reset to 0000h when its value equals TPR1.
TMR2	2	2	Timer 2. 16-bit timer that can be used as an event counter. TMR2 is reset to 0000h when its value equals TPR2.
TPR2	3	2	Timer 2 period register. Causes TMR2 to reset to 0000h when its value equals TPR2.
TCON	4	2	Timer control register. Controls operation and configuration of Timers 1 and 2, and the compare and capture subsystems.
Event Manager			
CMPR0 CMPR1 CMPR2 CMPR3 CMPR4 CMPR5	0 1 2 3 4 5	3 3 3 3 3 3	Compare registers. Contents of compare registers are constantly being compared with Timer 1 or Timer 2. When any one of the compare registers matches the timer, it generates an action specified by an action register. In the PWM mode, a match with the timer resets the corresponding CMPx pin to a low level.
ACT0 ACT1 ACT2 ACT3 ACT4 ACT5	0 1 2 3 4 5	4 4 4 4 4 4	Action registers. Contents of action registers determine what action, including generation of an interrupt, should take place on the CMPx pin when the compare registers match the timer. In the PWM mode, the action registers act as double buffers for the corresponding CMPRx register.
FIFO0 FIFO1 FIFO2 FIFO3	0 1 2 3	6 6 6 6	Four-deep FIFO stacks that capture timer values when a transition is detected on the corresponding CAPx pin.
CCON	4	6	Capture control register Controls configuration and operation of capture inputs. It also holds the status of the FIFOs.
CCLR	5	6	CCON bit clear register Allows clearing of individual bits in CCON without affecting other bits.
Serial Port			
SCON	0	5	Serial port control register. Controls configuration and operation of serial port.
SSET	1	5	SCON bit set register Allows setting of individual bits in SCON without affecting other bits.
SCLR	2	5	SCON bit clear register Allows clearing of individual bits in SCON without affecting other bits.

Table 3–10. Peripheral Registers(Continued)

Register	Port	Bank	Description
TBR	3	5	Transmit buffer register. Stores temporary data while old data is being shifted out from transmit register.
TSR	3	7	Transmit shift register. Stores outgoing data currently being transmitted.
RBR	4	5	Receive buffer register. Stores temporary data while new data is being shifted into receive register.
RSR	4	7	Receive shift register. Stores incoming data currently being received.
SMAT	2	7	Serial port match word register. Serial port stays in sleep mode until match of received value with contents of SMAT register is detected.
SBRG/ TPR3	5	5	Serial port baud rate generator. Sets divide ratios for serial port timer to generate baud rates in asynchronous mode. Can also be used as period register when not used by serial port.
STMR/ TMR3	5	7	Timer 3. Free-running 16-bit timer used for baud-rate generation for serial port. Can be used as general-purpose timer when not used by serial port.

3.9 Bit Selectable I/O Port (TMS320C14)

The TMS320C14 incorporates a 16-bit I/O Port (IOP) consisting of 16 individually bit-selectable I/O pins IOP0 (LSB) through IOP15 (MSB). Key features of the IOP are listed below:

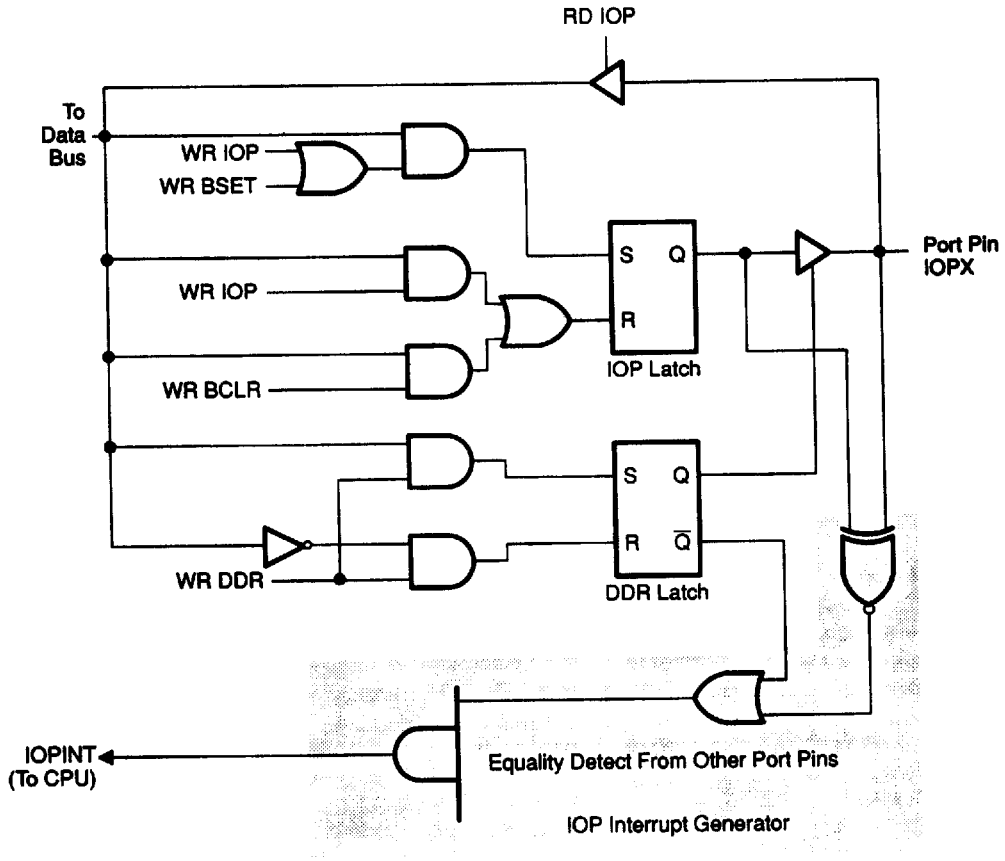
- 16 bit-selectable I/O pins
- Independent input/output pin configuration
- Independent set/clear control
- Dedicated register for output data storage
- Specific pattern detection
- Maskable interrupt

The IOP is controlled by registers configured by user software. Table 3–11 provides a summary of the IOP registers.

Table 3–11. I/O Port Register Summary

Register	Port	Bank	Description
IOP	0	0	I/O port latch. Stores data for IOP pins that are configured as output. Also stores data for pattern match on input IOP pins.
DDR	1	0	Data direction register. Configures IOP pins as inputs or outputs. DDR=1 configures the IOP pin as output. DDR=0 configures the pin as an input.
BSET	2	0	Bit set register. Allows setting of individual bits in IOP latch without affecting others. BSET=1 sets the IOP bit to a 1. BSET=0 leaves the IOP bit unaffected.
BCLR	3	0	Bit clear register. Allows clearing of individual bits in IOP latch without affecting others. BCLR=1 clears the IOP bit to a 0. BCLR=0 leaves the IOP unaffected.

Figure 3–34. Bit Selectable I/O Port



3.9.1 Configuring Data Direction

The direction of data at the IOP pins is determined by the 16-bit data direction register (DDR). Each bit of the DDR controls a corresponding pin in the IOP port. Each pin of the IOP can be individually configured as an input or output pin by specifying the corresponding bit in the DDR (see Figure 3–34). Clearing the bit to 0 configures the corresponding pin to be an input. Setting the bit to 1 configures the pin to be an output. Bit DDR0 configures pin IOP0, while bit DDR15 configures pin IOP15. Upon reset, the DDR is configured to be 0000h, thus making the I/O pins inputs. The DDR has a bank address of 0h and a port address of 1h.

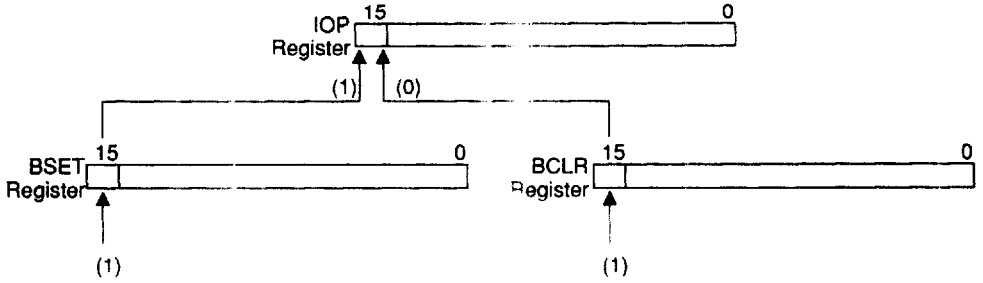
3.9.2 I/O Port Status and Control

A 16-bit latch called the IOP register is associated with the IOP port. The IOP register does the following:

1) Stores data to be transmitted on IOP pins configured as outputs.
2) Stores the status of IOP pins configured as inputs, when read, and a pattern to be matched with the input signals, when written.

Two additional 16-bit registers, the bit set (BSET) and bit clear (BCLR) facilitate the use of the IOP register. Each bit of these registers corresponds to a bit in the IOP register. Writing a one to bit 15 of the BSET register sets bit 15 of the IOP register. Writing a one to bit 15 of the BCLR register clears bit 15 of the IOP register (see Figure 3–35).

Figure 3–35. Configuring the IOP Register



Writing a zero to the BSET or BCLR has no effect on the IOP register. Reading the IOP register gives the status of the IOP pins configured as inputs, and the data to be transmitted on IOP pins configured as outputs. BSET register has a bank address of 0h and a port address of 2h. BCLR register uses bank address 0h and port address 3h. BSET and BCLR are write-only registers.

3.9.3 Input Pattern Match

The TMS320C14 provides an automatic compare feature. This feature uses IOP register bits that correspond to the IOP pins configured as inputs (see Figure 3–34). Because these bits would normally provide only the status of the input pins, the bits are available for an automatic compare function. A specific pattern can be stored in the IOP register of the bits corresponding to IOP inputs. This pattern is then constantly compared to the data received on the input pins. When a match occurs, an interrupt (IOPINT) is sent to the CPU. The pattern can be written to the IOP register directly or by using the BSET and BCLR register. Writing to the IOP register directly should be avoided, because this also affects other bits. Once a pattern is written into the IOP register, it cannot be read back. A read gives the data on the I/O port input pins, instead of the data in the IOP register bits.

The pattern in the IOP register must match all the pins configured as inputs. No subset of the input pins is possible. This feature helps the CPU detect a

comparison on the input ports without polling. The IOPINT interrupt appears only the first time a match is detected, after which it appears only if the comparison becomes false and then valid again. Interrupt IOPINT sets bit 0 in the interrupt flag register IF. IOPINT can be disabled by setting bit 0 in the interrupt mask register IM.

Note:

To reduce the risk of affecting the wrong bits, all writes to the IOP register should be through the BSET and BCLR registers.

3.10 Timers (TMS320C14)

The TMS320C14 is equipped with four independent timers: a watchdog timer, two general-purpose timers, and the internal clock/baud-rate generator. The first three 16-bit timers can be used as software development aids and also integrated into the DSP function. The fourth timer, the internal clock/baud rate generator, is intended for use with the serial port, but may also be used as a general-purpose timer if not required for serial communication.

Each of the four timers has these following features:

- A 16-bit free-running timer
- A 16-bit period register
- A dedicated maskable interrupt

Table 3–12. Timer Module Register Summary

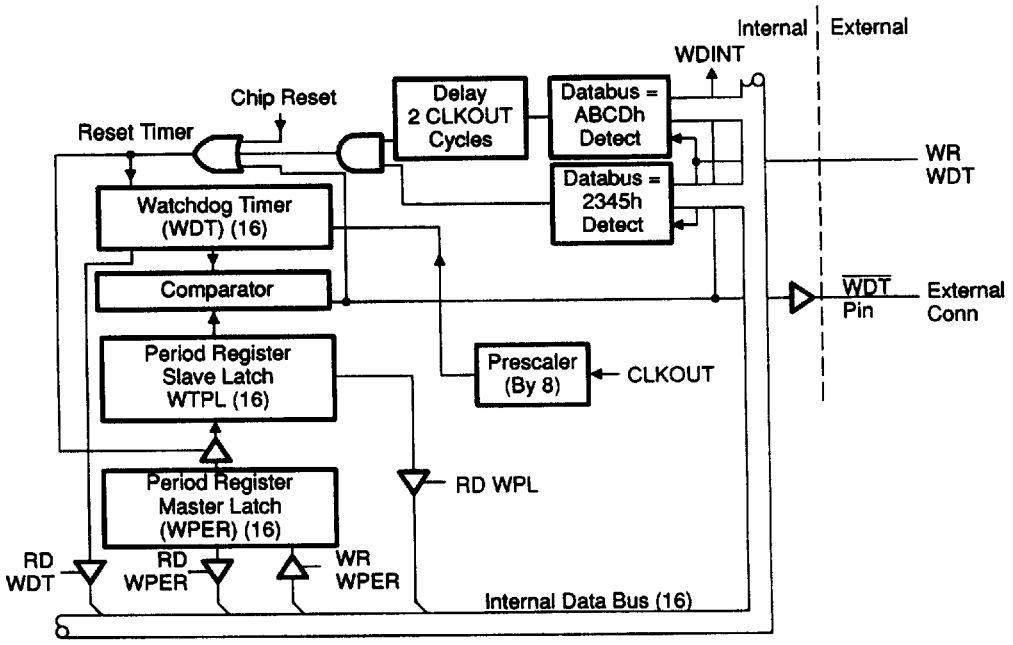
Register	Addr.	Bank	Description
WDT/TMR4	0	1	Watchdog timer/Timer 4. Free-running 16-bit timer that acts as a watchdog timer. Can also be used as a fourth timer. The timer is reset when it equals the period register.
WPER/TPR4	1	1	Watchdog timer period register. It causes the timer to be reset to 0000h when it equals the period register.
WTPL	2	1	Watchdog timer period register that prevents the watchdog period from being changed on the fly. This register stores the old value of WPER, used by WDT to determine its period.
TMR1	0	2	Timer 1. Free-running 16-bit timer that can be used as an event counter. It is reset to 0000h when its value equals TPR1.
TPR1	1	2	Timer 1 period register. It causes TMR1 to be reset to 0000h when its value equals TPR1.
TMR2	2	2	Timer 2. Free-running 16-bit timer that can be used as an event counter. It is reset to 0000h when its value equals TPR2.
TPR2	3	2	Timer 2 period register. It causes TMR2 to be reset to 0000h when its value equals TPR2.
TCON	4	2	Timer control register. Controls operation and configuration of Timers 1, 2, and event manager.
STMR/TMR3	5	7	Timer 3. Free-running 16-bit timer used for baud-rate generation for serial port. Can be used as general-purpose timer when not used by the serial port.
SBRG/TPR3	5	5	Baud rate generator. Sets divide ratios for serial port timer to generate baud rates in asynchronous mode. Can also be used as period register when not used by the serial port.

3.10.1 Watchdog Timer

The TMS320C14 is equipped with a free-running 16-bit watchdog timer (WDT). The timer module. shown in Figure 3–36 is designed to prevent software hang-ups. If the WDT is allowed to time out, a pulse is generated on pin WDT that can be used to reset the TMS320C14 and/or external hardware. In addition to the external pulse, an interrupt (WDTINT) is also routed to the CPU. In this way, a mechanism for recovering from software faults is provided.

To reset the watchdog timer (WDT), a pattern ABCDh followed by 2345h should be written to the WDT register with consecutive OUT instructions.

Figure 3–36. Watchdog Timer Module



The WDT is a read-only register and cannot be updated. However, WDT is reset to 0000h under the following conditions:

1) Pattern 0ABCDh followed by 02345h is written to the WDT by two consecutive OUT instructions. (Writing anything else has no effect.)
2) WDT times out when its contents match that of the period register latch WTPL.
3) The 'C14 is reset ($\overline{RS}$ driven low).

In the case of a timeout, interrupt WDTINT is generated, along with a pulse on pin WDT. If needed, the contents of the WDT can be read. WDT has a bank address of 1h and a port address of 0h.

3.10.1.1 WDT Period Register

Associated with the WDT are two 16-bit registers, WPER and WTPL. These registers operate in a master/slave relationship to store the maximum time that could occur between updates to the WDT. Timeout on the WDT occurs if it is not reset by consecutive writes, and its value matches with the contents of the WTPL.

The WPER is directly accessible from the data bus. It is double-buffered to prevent accidental changes to the watchdog period register. Associated with the WPER is a 16-bit buffer latch (WTPL) that stores the old value of the WPER if the WPER is changed. The WDT timer compares its value with the WTPL instead of the WPER. If the WPER is updated in the meantime, it has no effect on WTPL or on the period of the WDT. The WTPL is not directly accessible for writing and is updated only when the WDT is reset to 0000h. At that time, the new period value is transferred from the WPER to the WTPL. However, the WTPL can be read if needed. The WTPL uses bank address 1h and port address 2h. The WPER uses bank address 1h and port address 1h. At reset, the WPER is set to 0FFFFh (maximum timeout value).

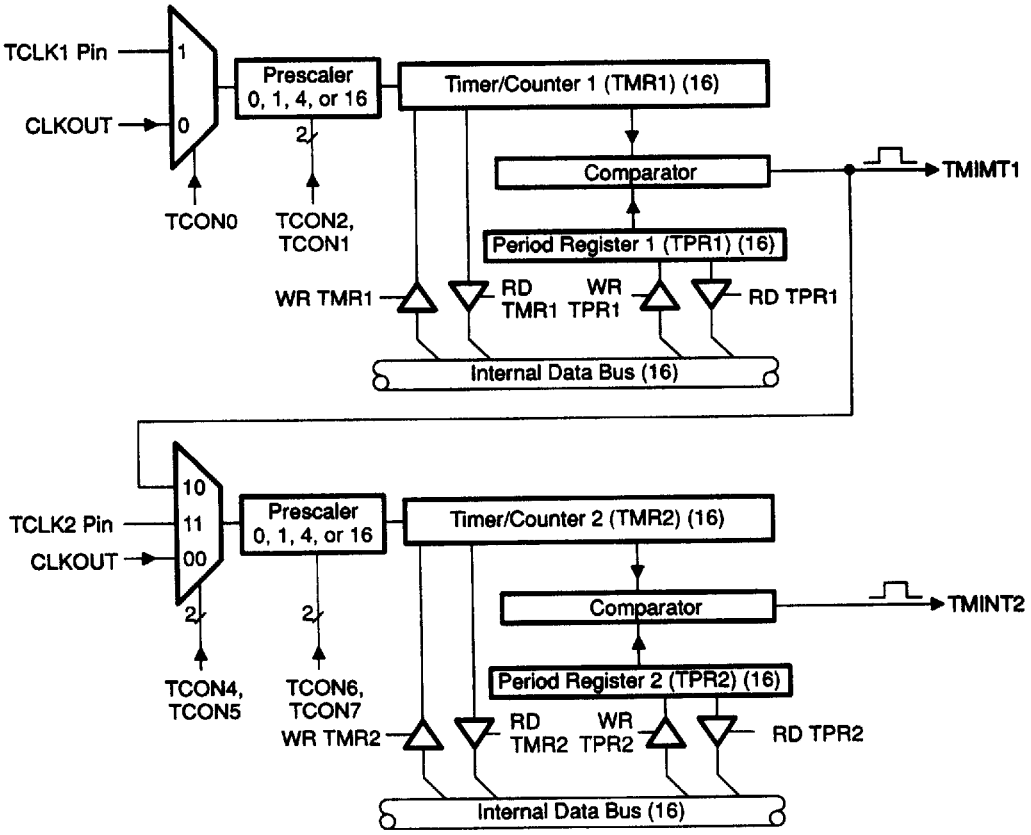
3.10.1.2 WDT Input

Input to the WDT is CLKOUT (CLKIN/4) divided by 8, for a total divide of 32 at 25.6 MHz, this gives a clock rate of 800 kHz. The external pulse generated on pin WDT when a timeout occurs is one WDT clock cycle or 8 CLKOUT periods. The maximum value that can be entered in the WPER is FFFFh, which gives a maximum period of 81.9 ms at 25.6 MHz. The interrupt routed to the CPU, WDTINT, sets bit 1 in the interrupt flag (IF) register. WDTINT can be disabled by setting bit 1 in the interrupt mask (IM) register. Although the WDT is not writable (except for the reset pattern), it can still be read and used as a regular timer in the system. The WDT can be used as a general-purpose timer if it is not being used for the watchdog function. Table 3–12 provides a summary of the registers used by the WDT.

3.10.2 General-Purpose Timers

The TMS320C14 is equipped with two 16-bit general-purpose incrementing timers (TMR1 and TMR2) that can be used as event counters (see Figure 3–37). Input to the timers can be from internal or external sources. Associated with the timers are two 16-bit period registers. When a timer value matches the value in the corresponding period register, the timer is reset in the next count cycle. A corresponding interrupt to the CPU is then generated.

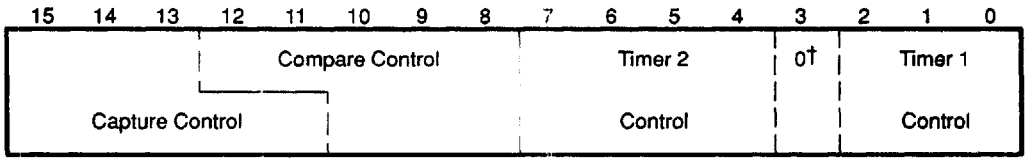
Figure 3–37. TMR1 and TMR2 Block Diagram



The operation of the timers is controlled by the timer control (TCON) register. Figure 3–38 shows the configuration of its bits. The TCON register also controls the compare and capture subsystems. It uses bank register 2h and port address 4h. Table 3–13 shows a summary of the TCON register bit functions.

Bit 0 of the TCON selects the clock source for TMR1. If bit 0 is 0, TMR1 input is from internal clock (CLKOUT). If bit 0 is 1, TMR1 input is from external source on pin TCLK1. The maximum clock rate from an external source is CLKOUT/2, where TMR1 is incremented on the rising edge of TCLK1. If desired, the input to TMR1 can be prescaled by bits 1 and 2, regardless of whether internal or external clock is selected. The clock input can have a prescale factor of 1 (no prescaling), 4, or 16. In addition, by setting both bits low (select 0) prescale bits 1 and 2 can also be used to stop or disable TMR1 at any time.

Figure 3–38. TCON Register Timer Bit Configuration



† Reserved bit. Should be cleared to 0.

Table 3–13. TCON Register Timer Description

Bit	Function
7, 6	Prescale select for TMR2. 00 = Timer stop. Stops timer as soon as set to 00. 01 = Prescale of 4. Divides input by 4. 10 = Prescale of 16. 11 = Prescale of 1. No prescaling. Timer disabled on reset.
5, 4	Selects clock source for TMR2. 00 = internal clock. 10 = TMR1 output is input clock. 11 = external clock.
3	Reserved. Should always be cleared (0).
2, 1	Prescale select for TMR1. 00 = Timer stop. Stops timer as soon as set to 00. 01 = Prescale of 4. Divides input clock by 4. 10 = Prescale of 16. Divides input clock by 16. 11 = Prescale of 1. No prescaling. Timer disabled on reset.
0	Clock select for TMR1. 0 = Internal clock (CLKOUT). 1 = External clock. Selects clock input on TCLK1.

Note: All bits cleared to 0 on reset.

Bits 4 and 5 specify the clock source for timer 2 (TMR2). In addition to internal (CLKOUT) and external clock, output of TMR1 can be specified as input to TMR2. It is thus possible to have a 32-bit timer with 8-bits of prescale. If an external clock is specified, input to TMR2 is from pin TCLK2. Bits 6 and 7 select the prescale values for TMR2 Bits 6 and 7 can also be used to hold or disable TMR2.

Both TMR1 and TMR2 can be written to and read from. TMR1 has a 16-bit period register (TPR1) associated with it. When the value of TMR1 matches TPR1, TMR1 is reset to 0000h, and an interrupt (TIMINT1) to the CPU is generated. TIMINT1 sets bit 4 in the interrupt flag (IF) register. TIMINT1 can be disabled by setting bit 4 in the interrupt mask (IM) register. TMR1 register has a bank address of 2h and a port address of 0h. TPR1 has a bank address of 2h and a port address of 1h. At reset TPR1 is set to FFFFh.

The TMR2 has a 16-bit period register (TPR2) associated with it. When the value of TMR2 matches TPR2, TMR2 is reset to 0000h, and an interrupt (TIMINT2) to the CPU is generated. TIMINT2 sets bit 5 in the IF register. It can be disabled by setting bit 5 in the IM register. TMR2 has a bank address of 2h and a port address of 2h. The TPR2 has a bank address of 2h and a port address of 3h. At reset, TPR2 is set to FFFFh.

3.10.3 Serial Port Baud Rate Generator (As a Timer)

The serial port baud rate generator (STMR) is primarily intended for use with the serial port during asynchronous communication. It can, however, be used as a 16-bit general-purpose timer if such communications are not used. The input to the STMR is always CLKOUT. Like the other timers, the STMR has a 16-bit period register (SBRG) associated with it, and an interrupt (STMRINT) can be generated to the CPU that sets bit 12 (which is maskable) in the interrupt flag (IF) register.

3.11 Event Manager (TMS320C14)

The TMS320C14 event manager consists of compare and capture subsystems. The event manager uses TMR1 and TMR2 as associated clock sources; its eight pins are divided between compare (CMP) and capture (CAP) subsystems.

The compare subsystem consists of compare and action registers and output pins. The compare registers compare their values with those of the timers. When a match is detected, the action registers specify an action that takes place on the output pins. The action registers can specify that an interrupt to the CPU should be generated. The compare subsystem also has a high precision PWM mode. In the PWM mode, six channels of high-precision PWM outputs are available.

The capture subsystem consists of four FIFOs and input pins. When a change is detected on the input pins, the current values of TMR1 and TMR2 are captured in the FIFOs. An interrupt to the CPU can also be generated at this time.

3.11.1 Compare Subsystem

The compare subsystem includes the following features:

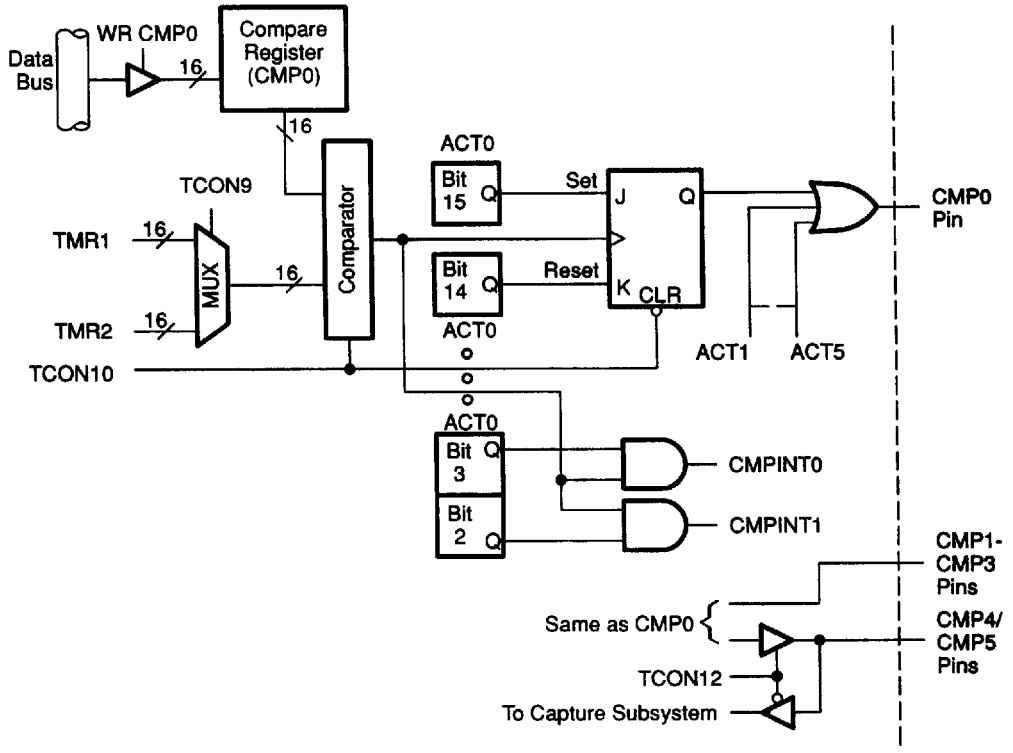
- Six 16-bit compare registers
- Six 16-bit action registers
- Six possible compare output registers
- High-precision PWM mode with double-buffered PWM registers and six PWM channels
- Two maskable interrupts

The 16-bit compare (CMPRx) registers are used for comparison with a selected timer, while the 16-bit action (ACTx) registers control the action of the output pins (see Figure 3–39). The action registers can specify generation of two interrupts (CMPINT0 and CMPINT1). The compare subsystem has four dedicated output pins (CMP0 through CMP3) and two pins that can be specified as compare outputs or capture inputs. Table 3–14 summarizes the registers associated with the compare subsystem.

Table 3–14. Compare Subsystem Register Summary

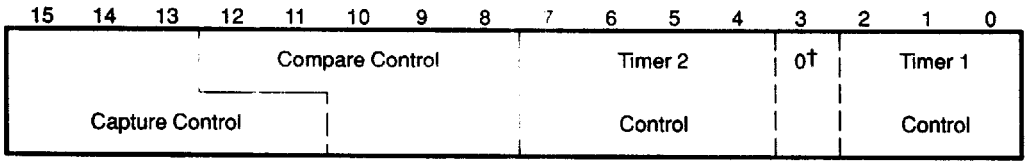
Register	Addr	Bank	Description
TCON	4	2	Timer control register. Controls operation and configuration of Timers 1 and 2 and the compare and capture subsystems.
CMPR0 CMPR1 CMPR2 CMPR3 CMPR4 CMPR5	0 1 2 3 4 5	3 3 3 3 3 3	Compare registers. Contents of compare registers are constantly being compared with Timer 1 or Timer 2. When any one of the compare registers matches the timer, it generates an action specified by an action register. In the PWM mode, a match with the timer resets the corresponding CMPx pin to a low level.
ACT0 ACT1 ACT2 ACT3 ACT4 ACT5	0 1 2 3 4 5	4 4 4 4 4 4	Action registers. Contents of action registers determine what action should take place on the CMPx pin when the compare registers match the timer, including generation of an interrupt. In the PWM mode, the action registers act as double buffers for the corresponding CMPRx register.

Figure 3–39. Compare Module



The compare registers are used to store certain values or periods. These are compared with timer values. Bits 8 through 12 of the timer control (TCON) register control the operation of the compare registers. Table 3–15 and Table 3–16 show the functions of the TCON register bits.

Table 3–15. TCON Register Compare Bit Configuration



† Reserved bit. Should be cleared to 0.

Table 3–16. TCON Compare Register Description

Bit #	Description
12	CMP5/CAP3 Configure. Set to zero on reset. 0 = Configures pin CMP5/CAP3 as capture input. 1 = Configures pin CMP5/CAP3 as compare output.
11	CMP4/CAP2 Configure. Set to zero on reset. 0 = Configures pin CMP4/CAP2 as capture input. 1 = Configures pin CMP4/CAP2 as compare output.
10	Compare Enable. Set to zero on reset. 0 = Disables the compare subsystem. Pins CMP0–CMP5 are held at 0. Compare registers CMP0–CMP5 are reset and held at 0000h; compare interrupts are disabled. 1 = Enables the compare subsystem. Allows normal operation of compare subsystems
9	Timer select for compare subsystem. Set to zero on reset. 0 = Timer 1. Select Timer 1 for use in comparison. 1 = Timer 2. Select Timer 2 for use in comparison.
8	Compare mode selection. Set to zero on reset. 0 = Normal compare subsystem operation. 1 = High precision PWM mode.

3.11.1.1 Compare Registers

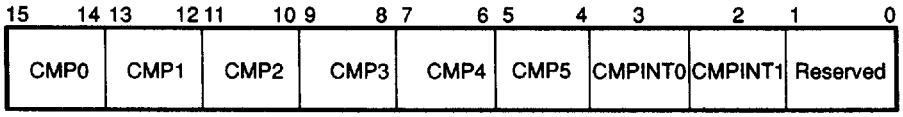
Once a value is stored in each of the six compare registers, the compare system works independently without any intervention from the CPU. Each of the compare registers (CMPR0 through CMPR5) is constantly comparing itself with either TMR1 or TMR2.

When a match is detected between the value stored in the compare register and the value in the timer, a match signal enables the corresponding action register to carry out an action on a compare output pin. All compare registers have the same bank address of 3h, and port addresses of 0h through 5h. They can be read from or written to using these addresses.

3.11.1.2 Action Registers

Each of the action registers is dedicated to a compare register, and is capable of controlling all six output pins (CMP0 through CMP5). When a match signal is received from the corresponding compare register (that is, value of timer matches compare register value), each of the compare output pins can either be set high, reset low, or toggled. In addition, each action register can specify generation of one or both interrupts (CMPINT0 and CMPINT1). Figure 3–40 shows the configuration of the bits in each action register.

Figure 3–40. ACTx Register Configuration



Bits 4 through 15 of the action register control (in pairs) the action of CMP0 through CMP5. Bits 2 and 3 control generation of CMPINT0 and CMPINT1. Bits 0 and 1 are reserved. ACT0 through ACT5 have the same bank address of 4h and port addresses 0h through 5h. They can be read from or written to via these addresses.

When the value in the specified timer equals the value in a compare register, the corresponding action register specifies an action, depending upon the configuration as shown in Table 3–17.

Table 3–17. Action Register Description

Bit #	Function	Description
15,14	Set/Reset CMP0	00 = No action taken on pin CMP0. 01 = Resets pin CMP0 to a low level. 10 = Sets pin CMP0 to a high level. 11 = Toggles pin CMP0.
13,12	Set/Reset CMP1	00 = No action taken on pin CMP1. 01 = Resets pin CMP1 to a low level. 10 = Sets pin CMP1 to a high level. 11 = Toggles pin CMP1.
11,10	Set/Reset CMP2	00 = No action taken on pin CMP2. 01 = Resets pin CMP2 to a low level. 10 = Sets pin CMP2 to a high level. 11 = Toggles pin CMP2.
9,8	Set/Reset CMP3	00 = No action on pin CMP3. 01 = Resets pin CMP3 to a low level. 10 = Sets pin CMP3 to a high level. 11 = Toggles pin CMP3.

Table 3–17. Action Register Description (Continued)

Bit #	Function	Description
7 6	Set/Reset CMP4	00 = No action taken on pin CMP4. 01 = Resets pin CMP4 to a low level. 10 = Sets pin CMP4 to a high level. 11 = Toggles pin CMP4.
5 4	Set/Reset CMP5	00 = No action on pin CMP5. 01 = Resets pin CMP5 to a low level. 10 = Sets pin CMP5 to a high level. 11 = Toggles pin CMP5.
3	Set CMPINT0	0 = No interrupt generated. 1 = Generates interrupt and sets bit 6 in IF register.
2	Set CMPINT1	0 = No interrupt generated. 1 = Generates interrupt and sets bit 7 in IF register.
1	Reserved	Should be set to 1.
0	Reserved	Should be set to 1.

3.11.1.3 Compare Pins

Four outpins (CMP0) through CMP3) are dedicated to the compare subsystem. Two additional pins are also available for the compare subsystem. These pins (CMP4/CAP2 and CMP5/CAP3) are shared with the capture (input) subsystem. CMP4/CAP2 and CMP5/CAP3 are configured by bits 11 and 12 of the TCON register. Each of the compare output pins can be controlled by all six action registers to create specific waveforms. However, if two action registers specify simultaneous action (that is, set is specified by one action register, while toggle is specified by another action register), unpredictable action can occur. Pins CMP0 through CMP3 are set low on reset. Pins CMP4/CAP2 and CMP5/CAP3 are configured as inputs and put in the high-impedance state at reset.

3.11.1.4 Compare Interrupts

Bits 2 and 3 of each action register can generate interrupts (CMPINT0 and CMPINT1) to the CPU when the corresponding compare register generates a match or EQ signal. CMPINT0 sets bit 6 in the interrupt flag register (IF). CMPINT1 sets bit 7 of IF. Both interrupts can be masked by using the interrupt mask register (IM).

3.11.1.5 High-Precision PWM Mode

The compare subsystem has a mode for generating high-precision pulse width modulation (PWM) outputs (refer to Figure 3–41). In the high-precision mode, the pulse width on pins CMPx has two extra bits of resolution. Thus, the pulse width in this mode can be specified with a minimum resolution of 40 ns @ 25.6 MHz (vs 160 ns in the normal compare mode). Figure 3–41 gives a comparison between the high-precision PWM mode and the normal compare mode.

Figure 3–41. Compare Subsystem in PWM

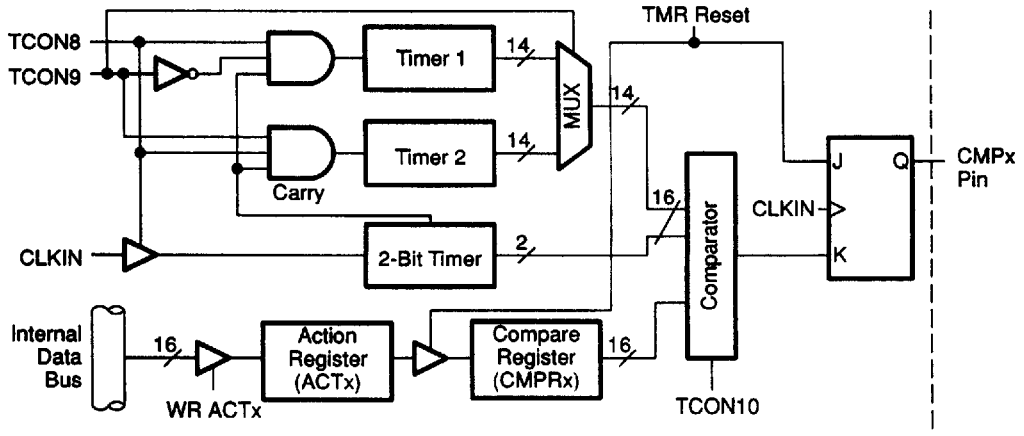


Table 3–18. PWM Resolution Bits Comparison

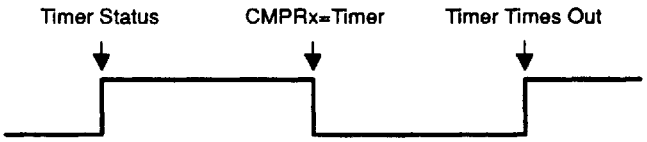
PWM Frequency (in KHz)	Bits of Resolution	
	Normal Compare Mode	High Precision PWM Mode
100	6	8
25	8	10
6.26	10	12
1.506	12	14

The PWM mode is enabled by setting bit 8 of TCON register to 1. In this mode, each of the six output pins is uniquely associated with one compare and action register, and each works independently. The pulse width of each compare output pin is determined by the associated compare register, while the overall period is determined by the selected timer period register.

To begin using the PWM mode, select TMR1 or TMR2 (prescale of 1, internal source) and load its period register with the period value. The selected timer, clocked by CLKOUT, begins to count (refer to Figure 3–42) until the value of the 14 LSBs of the timer match the 14 MSBs of the compare register. The two LSBs of the specified register are then used to count the number of CLKIN pulses before the associated compare pin is reset (refer to Figure 3–42). If the two LSBs are 00, a transition occurs immediately on the compare pin. If 01 appears on the two LSBs, a transition occurs after one CLKIN cycle. If 10 appears in the two LSBs, a transition occurs after two CLKIN cycles. If 11 appears in the two LSBs, a transition occurs after three CLKIN cycles. In this way, the resolution of the PWM mode is increased by a factor of four. The timer continues to count until its 14 LSBs match the 14 MSBs of its associated period register. When this occurs, all compare pins are set (refer to Figure 3–42), the timer is reset to 0000h, and the compare registers are loaded with the values in their associated action registers. Because only the 14 LSBs are used in comparing

the timer and its period register, the two MSBs in the period register ***must be zeros***. The timer begins counting from 0000h, and a new PWM cycle begins.

Figure 3–42. CMPx Pin Level

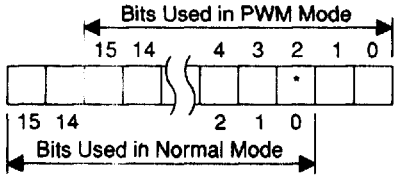


Note that the action of the CMPx pins are no longer controlled by the action registers. The action registers also do not generate any interrupts. Interrupt CMPINT0 is dedicated to CMPR4, and is generated when contents of CMPR4 match contents of TMRx. Interrupt CMPINT1 is dedicated to CMPR5, and is generated when contents of CMPR5 match contents of TMRx. Either TMR1 or TMR2 can be used for comparison in this mode. Note that the selected timer must be clocked at the CLKOUT rate with a prescale of one. See Figure 3–43 for TMR bit configuration.

In summary:

1) TMR1 or TMR2 is selected and the associated period register is loaded with the pulse period value.
2) TMRx counts until its 14 LSBs match the 14 MSBs of the compare register.
3) The two LSBs of the compare register decide which quarter phase causes the compare pin to reset.
4) When contents of register CMPR4 matches value of TMRx, pin CMP4 is reset to 0 and interrupt CMPINT0 is generated.
5) When contents of register CMPR5 matches value of TMRx, pin CMP5 is reset to 0 and interrupt CMPINT1 is generated.
6) TMRx continues to count until its 14 LSBs match the 14 LSBs of its associated period register. Recall that the two MSBs in the period register must be zeros.
7) ALL compare pins are set.
8) TMRx is reset to 0000h and begins counting.
9) The value in the action register is automatically written into its associated compare register.
10) New PWM cycle starts.

Figure 3–43. TMR Bit Configuration



3.11.2 Capture Subsystem

The capture subsystem provides a logging function for up to four different events (transitions) and includes the following features:

- Four 16 × 4 FIFO stacks
- Four possible Schmidt trigger input capture pins
- User-specified edge detection
- FIFO status indicator bits
- Optional timer selection

The operation of the capture subsystem is controlled by the timer control (TCON) and the capture control (CCON) register. Figure 3–44 shows the structure of the capture subsystem. Table 3–19 summarizes the registers associated with the capture subsystem.

Figure 3–44. Capture Module

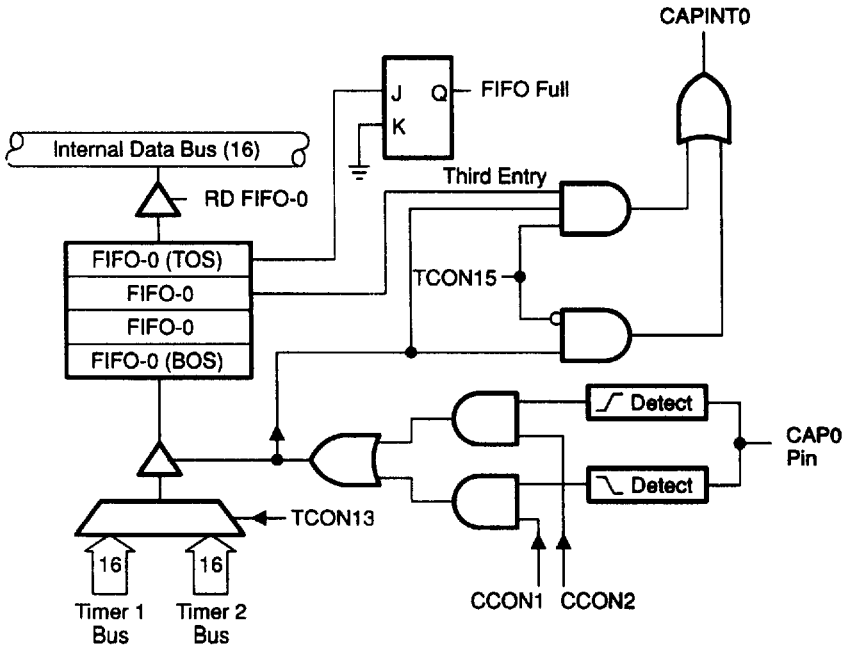


Table 3–19. Capture Subsystem Register Summary

Register	Addr	Bank	Description
TCON	4	2	Timer control register. Controls operation and configuration of TMR1, TMR2, and the compare and capture subsystems.
FIFO0 FIFO1 FIFO2 FIFO3	0 1 2 3	6 6 6 6	Four-deep FIFO stacks that capture the timer values when a transition is detected on the associated CAPx pin.
CCON	4	6	Capture control register. Controls configuration and operation of capture inputs. It also holds the status of the FIFOs.
CCLR	5	6	CCON bit clear register. Allows clearing of individual bits in CCON without affecting other bits.

Anytime a positive or negative transition is detected on a capture input pin, the current value of a timer is saved in a FIFO. In addition, when a capture is made, each FIFO can generate an interrupt to the CPU. There are four interrupts (CAPINT0 through CAPINT3) dedicated to the capture subsystem.

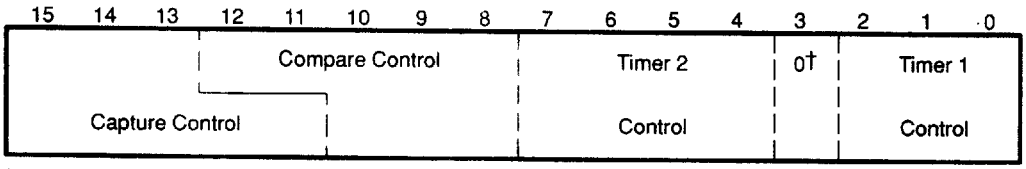
3.11.2.1 FIFO Stacks

Each of the FIFOs is four deep and dedicated to one of the capture input pins. When a transition is detected on a capture input pin, the current value of the timer is saved in the corresponding FIFO. Either TMR1 or TMR2 may be specified. Each FIFO has a dedicated interrupt (CAPINTx). The FIFO can generate an interrupt when the first capture is made or when the third capture entry is received. The CPU has direct access only to the top of each FIFO. The rest of the FIFO entries must be read successively. When a FIFO is empty (that is, all entries are read), a status bit (FIFOx not empty) is cleared in CCON. When a FIFO is full and an additional capture is made, the overrun condition is indicated by setting a status bit (FIFOx overrun) in CCON. At the same time, the FIFO preserves existing entries and ignores additional entries, thus losing them. The FIFO registers have a bank address of 6h, and a port address of 0h through 3h.

3.11.2.2 TCON Register

Bits 11 through 15 of the timer control (TCON) register control the operation of the capture subsystem. Figure 3–45 and Table 3–20 show the functions of the TCON register bits associated with the capture subsystem.

Figure 3–45. TCON Register Capture Bit Configuration



† Reserved bit. Should be cleared to 0.

Table 3–20. TCON Capture Register Description

Bit #	Description
15	Interrupt on first or third entry in FIFOx. 0 = An interrupt is generated to the CPU on the first capture entry received in an empty FIFO. 1 = An interrupt is generated to the CPU when the third capture entry is received in the FIFO and the FIFO already has two entries.
14	Capture Enable. 0 = Enables the capture subsystem. Starts normal operation of capture subsystem. 1 = Disables the capture subsystem. No capture is possible on any pin. All FIFO registers (FIFOx) are reset to 0000h. All FIFOx not empty bits are reset to 0. All FIFOx overrun bits reset to 0.
13	Timer select for capture subsystem. 0 = Timer 1. Selects Timer 1 for use in capture. 1 = Timer 2. Selects Timer 2 for use in capture.
12	CMP5/CAP3 Configure. 0 = Configures pin CMP5/CAP3 as capture input. 1 = Configures pin CMP5/CAP3 as compare output.
11	CMP4/CAP2 Configure. 0 = Configure pin CMP4/CAP2 as capture input. 1 = Configure pin CMP4/CAP2 as compare output.

Note: All bits cleared to zero on reset.

3.11.2.3 Capture Control Register

The capture control register, CCON, (refer to Figure 3–46 and Table 3–20) is used to configure and control the operation of individual capture inputs. Each capture input has four bits in CCON that controls its operation. Capture can be enabled or disabled on each individual capture pin. In addition, each pin can be programmed to detect a transition, a falling edge, or a rising edge. If either CMP4/CAP2 or CMP5/CAP3 pins are configured as output compare pins, it is still possible to enable a capture function on these pins. In this case, the transition detected is due to the compare output function, not an external input.

The CCON gives the status of each of the FIFOs, whether they are empty or have an overflow. When an overflow is detected and bit FIFOx overrun is set to 1, CCON must be cleared to 0. You can do this in software after a read from FIFOx is completed. Some bits of the CCON, that is, FIFO status bits, can only be read only while the rest of the bits can be written to and read from.

Another 16-bit register, capture clear (CCLR), is also used by the capture subsystem. The CCLR allows clearing of individual bits in CCON without affecting other bits. The operation of CCLR is similar to that of the BCLR register. There is a direct correlation between bits of CCLR and CCON that is: setting bit 3 of CCLR to 1 clears bit 3 of CCON. Figure 3–46 illustrates bit 15 of CCLR clearing 15 of CCON. The CCLR makes it easier to configure the CCON because it is not possible to write to CCON directly. The CCON has a bank address of 6h

and port address of 4h. The CCLR has a bank address of 6h and port address of 5h and is described in Table 3–21.

Figure 3–46. CCON and CCLR Register

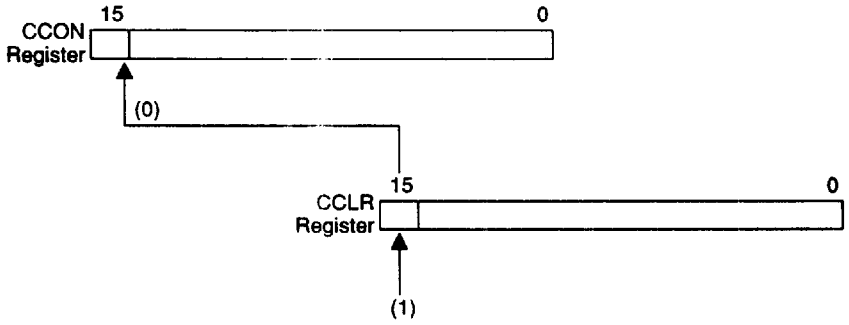


Table 3–21. CCON Register Description

Bit #	Function	Description
15	FIFO-3 overrun error.	0 = FIFO-3 is not full. 1 = FIFO-3 is full, and another transition or capture entry is detected. You must clear this bit if a read is done on FIFO-3. Status bit. Read and clear access.
14	FIFO-3 not empty.	0 = No entries available in FIFO-3. 1 = FIFO-3 has one or more unread entries. Cleared as soon as the CPU has read all entries and FIFO-3 is empty. Status bit. Read access only.
13.12	Edge transition detection for CAP3.	00 = Capture disable on pin CAP3. 01 = Detect positive (rising) edge on pin CAP3. 10 = Detect negative (falling) edge on pin CAP3. 11 = Detect any transition on pin CAP3. Control bits. Read and write access.
11	FIFO-2 overrun error.	0 = FIFO-2 is not full. 1 = FIFO-2 is full, and another transition or capture entry is detected. You must clear this bit if a read is done on FIFO-2. Status bit. Read and clear access.
10	FIFO-2 not empty.	0 = No entries available in FIFO-2. 1 = FIFO-2 has one or more unread entries. Cleared as soon as CPU has read all entries and FIFO-2 is empty. Status bit. Read access only.
9, 8	Edge transition detection for CAP2.	00 = Capture disabled on pin CAP2. 01 = Detect positive (rising) edge on pin CAP2. 10 = Detect negative (falling) edge on pin CAP2. 11 = Detect any transition on pin CAP2. Control bit. Read and write access.

Table 3–21. CCON Register Description (Continued)

Bit #	Function	Description
7	FIFO-1 overrun error.	0 = FIFO-1 is not full. 1 = FIFO-1 is full, and another transition or capture entry is detected. You must clear this bit if a read is done on FIFO-1. Status bit. Read and clear access.
6	FIFO-1 not empty.	0 = No entries available in FIFO-1. 1 = FIFO-1 has one or more unread entries. Cleared as soon as the CPU has read all entries and FIFO-1 is empty. Status bit. Read access only.
5,4	Edge transition detection for CAP1.	00 = Capture disabled on pin CAP1. 01 = Detect positive (rising) edge on CAP1. 10 = Detect negative (falling) edge on pin CAP1. 11 = Detect any transition on pin CAP1. Control bits. Read and write access.
3	FIFO-0 overrun error.	0 = FIFO-0 is not full. 1 = FIFO-1 is full, and another transition or capture entry is detected. You must clear this bit if a read is done on FIFO-0.
2	FIFO-0 not empty.	0 = No entries available in FIFO-0. 1 = FIFO-0 has one or more unread entries. Cleared as soon as CPU has read all entries and FIFO-0 is empty. Status bit. Read access only.
1,0	Edge transition detection for CAP0	00 = Capture disabled on pin CAP0. 01 = Detect positive (rising) edge on CAP0. 10 = Detect negative (falling) edge on CAP0. 11 = Detect any transition on CAP0. Control bits. Read and write access.

Note: All bits cleared to zero on reset.

3.11.2.4 Capture Interrupts

Each capture input has a dedicated interrupt (CAPINTx) that can be generated either when the corresponding FIFO receives either its first or third entry. This selection is done by means of bit 15 of TCON. CAPINT0 through CAPINT3 set bits 8 through 11 of IF. These interrupts can be masked by setting the appropriate bit in IM.

3.12 Serial Port (TMS320C14)

The TMS320C14 has a universal asynchronous receiver/transmitter (UART) serial port that supports industry-standard communications protocols. Key features of the serial port include the following:

- Double-buffered receiver/transmitter.
- Full-duplex asynchronous mode with 400 Kbps maximum transceiving rate (25.6-MHz clock).
- Supports address detect and address match protocols.
- Separate interrupts for receiver/transmitter.
- Separate 16-bit timer for baud rate generation.
- Programmable operation through 16-bit control/status register.

The asynchronous (async) mode is full-duplex with a maximum transmission/reception rate of CLKOUT/16. The async mode uses start and stop bits for synchronization at the byte level. Double-buffering of receive/transmit data is used in all modes.

The serial port supports communication protocols. The first communication protocol supports 8051 nine-bit communication, in which the ninth bit indicates address or data reception/transmission. The second communication protocol supports a sleep/wakeup mode, in which all nine bits have to match the device address stored in a register (SMAT) to start data reception. Table 3–22 summarizes the registers associated with the serial port.

Table 3–22. Serial Port Register Summary

Register	Addr	Bank	Description
SCON	0	5	Serial port control register. Controls configuration and operation of the serial port.
SSET	1	5	SCON bit set register. Allows setting of individual bits in SCON without affecting other bits.
SCLR	2	5	SCON bit clear register. Allows clearing of individual bits in SCON without affecting other bits.
TBR	3	5	Transmit buffer register. Stores temporary data while old data is being shifted out from transmit register.
RBR	4	5	Receive buffer register. Stores temporary data while new data is being shifted into receive register.
SBRG	5	5	Serial port baud rate generator. Sets the divide ratios for the serial port timer to generate baud rates for asynchronous mode.
SMAT	2	7	Serial port match word register. The serial port stays in the sleep mode until an address match with value in SMAT register is detected.
TSR	3	7	Transmit shift register. Stores outgoing data that is currently being transmitted.
RSR	4	7	Receive shift register. Stores incoming data that is currently being received.
STMR/TMR3	5	7	Timer 3. Free running timer that is used for baud rate generation for the serial port.

3.12.1 Serial Control Register

The serial control (SCON) register see Table 3–23) is a 16-bit register that controls operation of the serial port. At the same time, the SCON also acts as a status register to indicate the status of the serial port. Async mode is selected by configuring bits 0 and 14 of the SCON. Reception on the serial port can be disabled by bit 11 of the SCON. Bits 5 and 6 of SCON select the number of data bits, while bits 9 and 10 select the communication protocol. SCON bits 3,4,7, and 8 indicate errors or overflows. The SCON should not be modified during reception or transmission because unpredictable results may occur. The complete configuration of the SCON is described in the following section. The SCON has a bank address of 5h and a port address of 0h.

Table 3–23. SCON Register Description

Bit #	Function	Description
0	Sync mode (SYNC)	0 = Selects asynchronous mode. Note that in addition to configuring this bit, bit 14 also has to be cleared to 0 to select async mode. Set to 1 on reset. Control bit. Write and read access. Must be cleared after reset to operate the serial port. 1 = Reset condition (serial port is nonoperational).
1	Parity enable (PEN)	0 = Disable parity. 1 = Enables parity. Parity is added during transmission, and detected during reception. Control bit. Write and read access. Cleared to 0 on reset.
2	Even/Odd parity select (EOP)	0 = Selects even parity. 1 = Selects odd parity. Control bit. Write and read access. Cleared to 0 on reset.
3	Parity error detect	0 = Normal operation. No parity error detected. 1 = Parity error has been detected during reception. You must clear it in software. Status bit. Read and clear access only. Cleared to zero on reset.
4	Framing error detect (FERR)	0 = Normal operation. No framing error detected. 1 = Framing error has been detected, that is, invalid stop bit received. You must clear it in software. Status bit. Read or clear access. Cleared to 0 on reset.
6,5	Data bits select (DN0, DN1)	00 = Selects 6 data bits. 01 = Selects 7 data bits. 10 = Selects 8 data bits. 11 = Selects 9 data bits. Control bits. Write and read access. Set to 10 on reset.
7	Receiver overflow error (ROV)	0 = Normal operation. No overflow detected. 1 = Receiver buffer overflow detected. Both receive registers (RSR and RBR) are full, and third data reception is detected. You must clear it in software. Status bit. Read and clear access only. Cleared to zero on reset.
8	Receive buffers full (BBF)	0 = Normal operation. Receive buffers empty. 1 = Receive buffers full detected. Both receive registers/buffers (RSR and RBR) have just filled. New data word is fully received in RSR. Status bit. Read and clear access. Control bit. Cleared to 0 on reset.
9	Receive interrupt qualifier1 (RINTQ1)	0 = Protocol 1 disabled. Normal reception mode. Receive interrupt (RXINT) generated to CPU every time new data frame is clocked into RSR register. 1 = Interrupt CPU if MSB of receive word is 1. Protocol 1 is enabled, and normal reception mode is disabled. Serial port is put into sleep mode. Receive interrupt (RXINT) is generated if MSB or last bit of received word is 1 (indicating address reception) Control bit. Write and read access. Cleared to zero on reset.
10	Receive interrupt qualifier2 (RINTQ2)	0 = Protocol 2 disabled. Normal reception mode. Receive interrupt (RXINT) generated to CPU every time new data is clocked into RSR register. 1 = Interrupt CPU if receive word matches SMAT. Protocol 2 is enabled, and normal reception mode is disabled. Serial port is put into sleep mode. Receive interrupt (RXINT) is generated if received word matches value in SMAT (indicates serial port has been addressed). Control bit. Write and read access. Cleared to zero on reset

Table 3–23. SCON Register Description (Continued)

Bit #	Function	Description
11	Continuous receive enable (CREN)	0 = Disables reception 1 = Enables reception/continuous reception. Control bit. Write and read access. Set to 1 on reset.
12	Single receive enable (SREN)/Frame sync pulse width (FSPW)	0 = Sets frame sync pulse generated by TMS320C14 at one clock pulse wide. Can be overridden by CREN bit. 1 = Sets frame sync pulse generated by the TMS320C14 at two pulses wide. Control bit. Write and read access. Cleared to 0 at reset
13	Reserved	Cleared to 0 on reset.
14	Asynchronous mode	Must be zero to operate the serial port in asynchronous mode. Control bit. Write and read access. Cleared to 0 on reset.
15	Transmit buffer empty (TBE)	0 = Transmit buffer not empty. Transmit buffer contains data that has not been transferred to transmit shift register. 1 = Transmit buffer empty. Transmit buffer is empty and ready to accept new data from the CPU. Status bit. Read access only. Set to 1 on reset.

To facilitate clearing and setting of individual bits in the SCON, two additional registers (SCLR and SSET) are provided (see Figure 3–47). SCLR and SSET function similarly to BCLR and BSET. They clear or set individual bits in the SCON. A mask of ones written to SCLR clear the corresponding bits in the SCON. Bit positions in SCLR containing zeros do not affect corresponding bits in the SCON. Similarly, SSET sets to one all bits in the SCON that have corresponding ones in the SSET. Bit positions having zeros do not affect corresponding bits in the SCON. The SCLR and SSET have a bank address of 5h and port address of 1h and 2h.

Figure 3–47. SCON Register Control

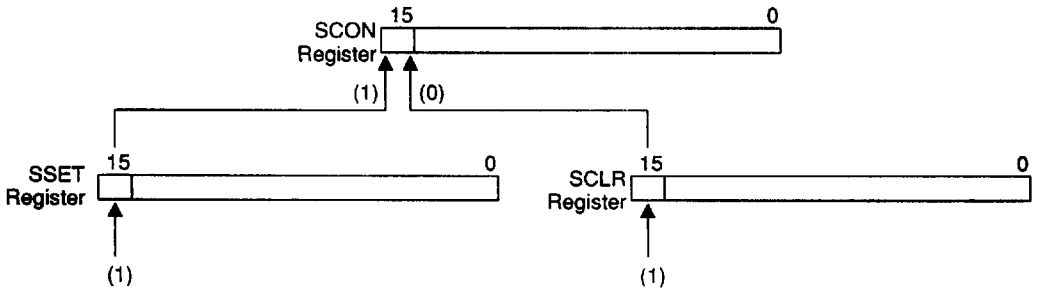


Table 3–24 shows the basic parameter configuration of the serial port after power-up or a reset.

Table 3–24. Serial Port Key Default Settings

Parameter	Description
Mode	Nonoperational
Parity	None
Word Length	8 Data bits
Reception	Continuous
Clock	Internal

3.12.2 Serial Port Baud Rate Generator

The serial port baud rate generator provides internal baud rate generation for asynchronous mode. If the baud rate generator is not required for clock/baud rate generation, it can be used as a general-purpose timer (refer to subsection 3.5.3) with its own interrupt. When the baud rate generator is used for baud rate/clock generation, bit 12 of the interrupt mask (IM) register should be masked.

The baud rate generator has a 16-bit register (STMR) and a baud rate generation register (SBRG) that uses CLKOUT as the input; see Figure 3–48. The SBRG register holds a user-installed value that is used in generating the baud rate.

3.12.2.1 Asynchronous Baud Rate Generation

The asynchronous baud rate is computed as follows:

$$\text{BaudRate} = \frac{\text{CLKOUTfrequency}}{16(K + 1)}$$

where: K = value stored in the SBRG register,
and CLKOUT = CLKIN/4

The maximum transmission/reception rate is CLKOUT/16, or 400 kHz at 25.6 MHz CLKIN frequency. Table 3–25 gives the K values that have to be stored in the SBRG register to obtain the standard baud rates and the deviation from those rates. These calculations use the maximum oscillator frequency of 25.6 MHz.

Table 3–25. SBRG Value for Standard Baud Rates

Desired Baud Rate	SBRG Value (K)	Actual Baud Rate
19.2 Kbps	0014h	19.048 Kbps
9600 bps	0028h	9756.00 bps
4800 bps	0052h	4819.00 bps
2400 bps	00A6h	2395.20 bps
1200 bps	014Ch	1201.20 bps
300 bps	0534h	300.07 bps
110 bps	0E33h	110.011 bps

Note: All K values are in hexadecimal.

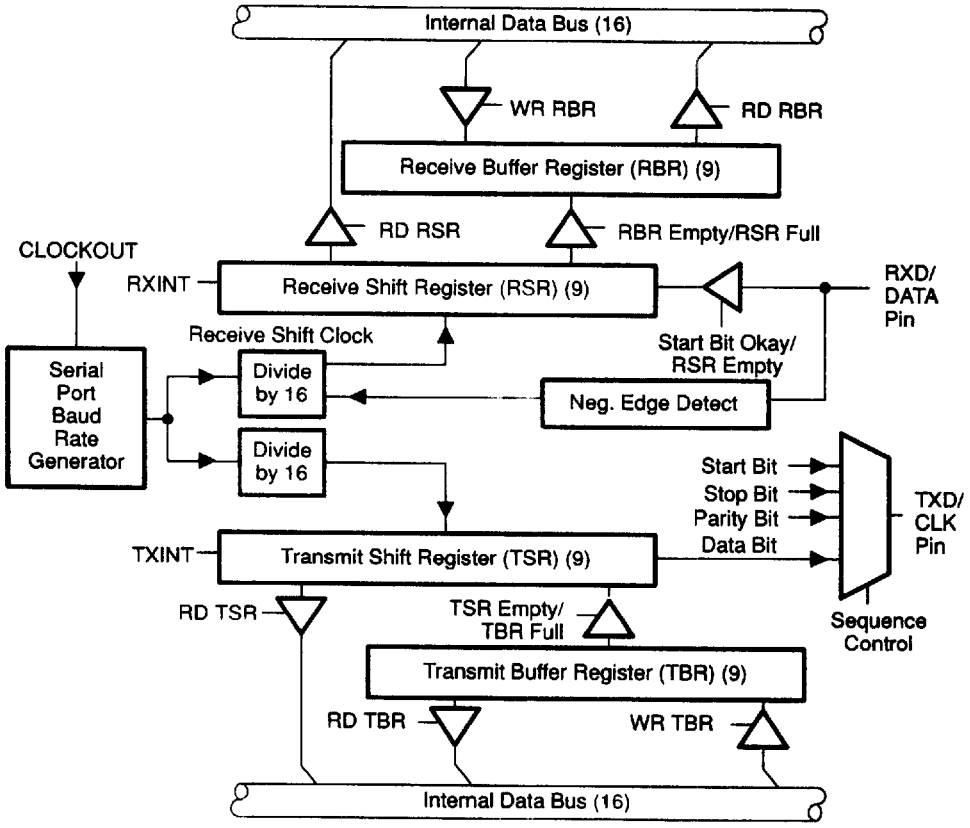
The maximum baud rate is obtained when K = 0. The minimum baud rate is generated when the value 65535 (FFFFh) is stored in the SBRG register. The percentage deviation (Pcd) is computed as follows:

Pcd = (Actual rate – desired rate)/desired rate

3.12.3 Asynchronous Mode

The asynchronous mode is selected by clearing bits 0 and 14 of the SCON to 0. This is a full duplex mode, with data being transmitted on the TXD/CLK pin and data received on the RXD/DATA pin. Figure 3–48 shows the architecture of the serial port for asynchronous communication.

Figure 3–48. Asynchronous Serial Port Operation

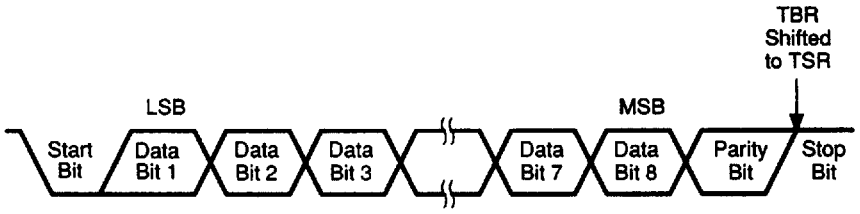


In the asynchronous mode, data is transmitted on a character-by-character basis with each data frame containing a start bit, 6 to 9 data bits, a parity bit (if parity is enabled), and a stop bit. Both the transmit and receive sections are double-buffered to allow continuous transceiving. The serial port timer provides the transmit and receive register clock signals, as well as the baud rate.

3.12.3.1 Asynchronous Transmission

The transmit section consists of a 9-bit transmit shift register (TSR) and a 9-bit transmit buffer register (TBR). Data is always written to the TBR, and then transferred to the TSR. Data to the TBR should be written in right-justified form, irrespective of the number of the bits to be transmitted. It is possible to read the TSR directly, but not to write to it directly. Data from the TSR is shifted out on pin TXD/CLK. The TXD/CLK pin is held at 1 while the serial port is not transmitting. Figure 3–49 shows an asynchronous transmission of 8-bit data with parity bit.

Figure 3–49. Asynchronous Transmission of Eight Bits Plus Parity



Transmission in the serial port is started by writing data to the TBR. If the TSR is empty, data from the TBR is transferred to the TSR. If the TSR is full, then data is kept in the TBR, and existing data in the TSR is shifted out to the sequence control logic. At the same time, bit 15 in the SCON register is cleared to 0 to indicate that the TBR is not empty. If both the TSR and TBR are full and the CPU tries to write to the TBR, the write is not allowed, and existing data in both registers is maintained. The sequence control logic constructs the transmit frame by outputting a start bit followed by the data bits from the TSR, and a parity bit is then added (if required). As soon as the last data or parity bit has left the TSR, TXINT is generated to the CPU, indicating that a frame has been sent.

The TSR has a bank address of 7h and a port address of 3h. The TBR has a bank address of 5h and a port address of 3h.

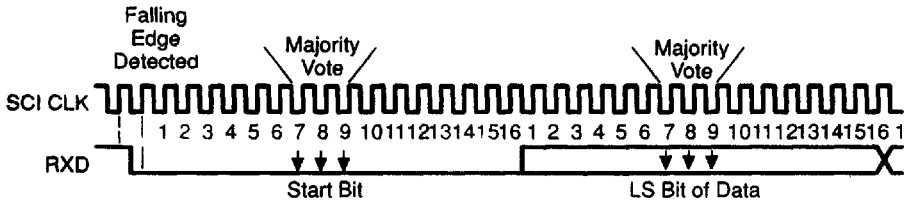
Summary of asynchronous mode transmission:

1) Asynchronous mode is enabled by clearing bit 0 of SCON to 0 and setting bit 11 to 1.
2) Transmission begins by writing data to TBR. If TSR is empty, this data is transferred to TSR.
3) Start bit is transmitted first, followed by 6 to 9 data bits (LSB through MSB).
4) If parity is enabled, parity bit is added after MSB.
5) Stop bit is shifted out and TXINT is generated, indicating end of transmission.

3.12.3.2 Asynchronous Reception

The receive section includes two 9-bit registers: the receive buffer register (RBR) and receive shift register (RSR). Data is received on the RXD/DATA pin, and the negative edge detect logic samples the input for a start bit on the 7th, 8th, and 9th sampling pulse following a detected falling edge (see Figure 3–50).

Figure 3–50. Start Bit Detection

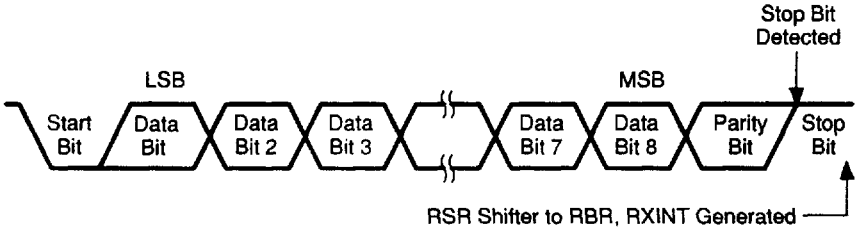


A majority vote of 0 by the three sample pulses results in the data being loaded into the RSR. If the majority vote is 1, the data is not accepted, and the sampling continues until a start bit is detected.

After the appropriate number of data bits is received (that is, 6 – 9 bits), parity is computed on the data bits (if parity is enabled) and compared with the received parity bit. If the computed parity does not match with the received, a parity error is indicated by setting a flag (bit 3 of SCON register – PERR), to a 1 and normal reception continues. You are responsible for checking this bit and clearing it. PERR (parity error) is also double buffered. This means when both buffers are full and PERR is raised, the data in RBR has a parity error. If data in RSR also has a parity error, PERR is raised again when data is transferred from RSR to RBR.

After the parity bit is received, a stop bit is received, indicating the end of that block. If a stop bit is not received, a framing error is indicated by the setting of bit 4(FERR) in SCON to a 1. Normal reception continues, and the receiver looks for the next start bit. You are responsible for checking this bit and clearing it. Data is then transferred to the RBR, and interrupt RXINT is routed to the CPU (see Figure 3–51). The RSR is now available to receive another data block, and the negative edge detect logic is activated again.

Figure 3–51. Asynchronous Reception of Eight Bits Plus Parity



If RBR is not empty when new data has just been received in the RSR, an error flag, BBF (both buffers full), is set in SCON. In this case, data from RSR is not transferred to RBR, and all further reception is disabled. Existing data is maintained in both RSR and RBR until you read RBR. Reading RBR automatically clears the BBF flag, and data in the RSR is transferred into RBR, thus allowing further reception. RSR has a bank address of 7h, and a port address of 4h. RBR has a bank address of 5h and a port address 4h.

Summary of asynchronous mode reception:

1) A negative edge is received to indicate a start bit. A test is performed to indicate whether or not a start bit is valid.
2) If start bit is valid, the appropriate number of data bits (as specified by the SCON register) is shifted into RSR.
3) (Optional). Parity is computed on the parity bits, and a parity bit is received. Computed parity is compared with the received parity. If they are different, a parity error is indicated.
4) A stop bit is received to indicate end of reception. If a stop bit is not received, a framing error is indicated.
5) An interrupt is generated to the CPU.
6) Data is transferred from RSR to RBR. If RBR is full, the BBF (both buffers full) flag is set.
7) Reception is complete; receiver waits for another negative transition.

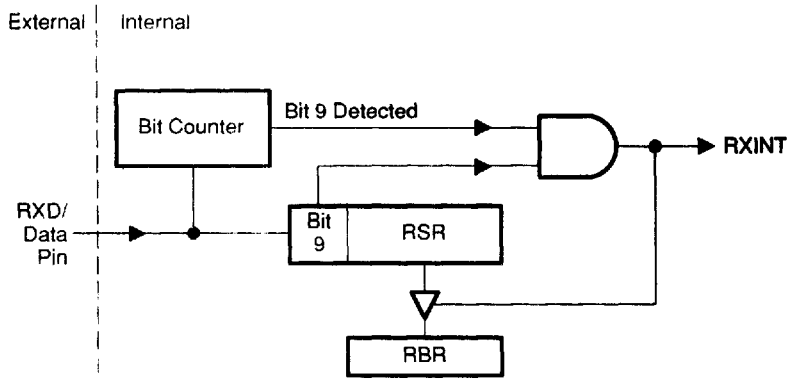
3.12.4 Communication Protocols

Besides supporting the regular transmission/reception (no protocol), the TMS320C14 supports two communications protocols: address detect and address match. These protocols are used for interprocessor communication and to allow the processor to ignore all reception until it is addressed by another device. The first communication protocol, address detect, allows the TMS320C14 to ignore all reception until it detects an address reception. The serial port then wakes up and allows you to determine via software if the right address was received or not. The second communication protocol, address match, allows the serial port to actually match the incoming address with its own address (stored in SMAT register). If the addresses match, the serial port wakes up. If not, all reception is ignored. The communications protocols are selected by bits 9 and 10 of the SCON register.

3.12.4.1 Address Detect Protocol

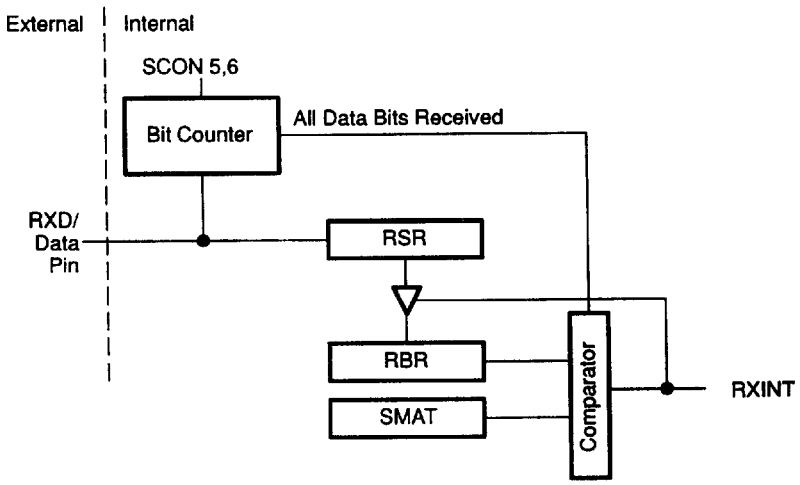
Address detect allows the processor to detect when an address is being received in the serial port. This protocol is enabled by setting RINTQ1 (receive interrupt qualifier 1), bit 9 of SCON, to 1. Address detect requires that 9-bit data transmission be used. The 9th or MSB is used to determine whether address or data is being transmitted. If the 9th bit equals 0, this indicates data is being received, and the serial port ignores the reception. Although data is being shifted into the RSR register, no interrupt (RXINT) is generated to the CPU. If the 9th bit equals 1, interrupt RXINT is generated, indicating that an address is being received (see Figure 3–52).

Figure 3–52. Serial Port Using Address Detect Protocol



Upon detection of an address, the serial port wakes up and generates interrupt RXINT to the CPU. Your software then matches the received address with its own address. If the address does not match, the serial port is allowed to remain in sleep mode. If a match is indicated, the software puts the serial port into wakeup by resetting RINTQ1, bit 9 of SCON to 0, disabling the communication protocol. This allows reception of data. When the complete message is received, RINTQ1 is set to 1, thus enabling the communication protocol and putting the serial port back into sleep mode. Data is shifted in with the rising edge of the signal. Every ninth clock is detected to determine whether the received bits are address or data (see Figure 3–53).

Figure 3–53. Address Detect Reception



When the address detect protocol is used, parity must be disabled. The receiver assumes that no parity is being sent along with the data.

Summary of address detect protocol:

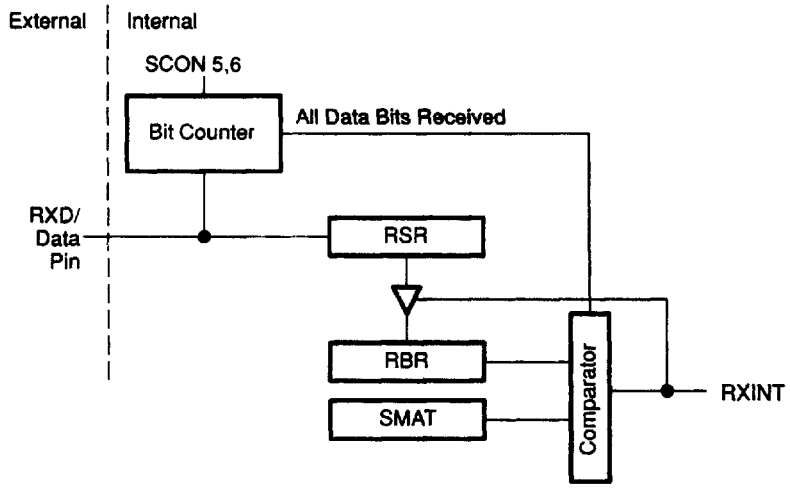
1) Nine data bits are selected, and parity is disabled.
2) Reception is disabled.
3) Setting RINTQ1 bit (bit 9 of SCON) to 1 selects protocol 1, and puts the serial port into sleep mode.
4) Data is clocked into the RSR register. The MSB (9th bit) is checked to see if it is a 1 or a 0.
5) If MSB is 0, the incoming word is assumed to be data, and nothing happens. The serial port continues in sleep mode.
6) If the MSB is 1, the incoming word is detected as address, and interrupt RXINT is generated to the CPU.
7) Your software looks at incoming address and decides whether to react or to continue sleeping.
8) If it decides to continue sleeping, nothing is done, and the serial port ignores all data clocked in, until another address is detected.
9) If you decide to wake it up, set RINTQ1 of SCON to 0.
10) Normal reception is enabled, and data is clocked into RSR, generating interrupt RXINT each time a word is shifted in.
11) You determine that a full message is received; set RINTQ1 to 1, to enable protocol and put the serial port back to sleep.

3.12.4.2 Address Match Protocol

Address match protocol allows the serial port to actually match an incoming address to determine if another device wants to communicate with it. This pro-

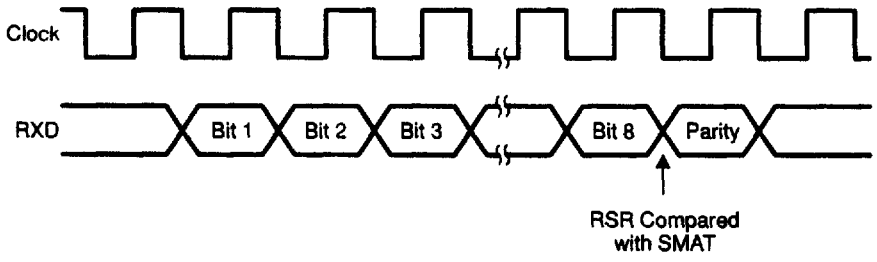
tocol is enabled by setting RINTQ2 (bit 10) of SCON register to a 1. Address match requires that an address be written to a 9-bit sync match register called SMAT. The incoming word is matched against the value in the SMAT register (see Figure 3–54 and Figure 3–55). If a match is detected, the serial port determines that it is being addressed and is activated. It then generates interrupt RXINT to the CPU. If no match is detected, the data clocked into the RSR is discarded, and the serial port remains in the sleep mode.

Figure 3–54. Serial Port Using Address Match Protocol



Once an address is detected, the serial port wakes up by generating an interrupt to the CPU. You must then put the serial port in the normal reception mode by resetting RINTQ2 (bit 10) of SCON to a 0. Data will now be clocked into the RSR, generating interrupt RXINT every time a complete word is received. When the complete message is received, you can put the serial port back into sleep mode by setting RINTQ2 back to a 1. Address match protocol does not require parity to be disabled, nor does it require that the full 9-bit data length be used. If the value written into the SMAT register is less than nine bits, the value must be right-justified, and the remaining upper bits must be zeros. Note that to generate a unique address, the number of bits in the address frame must exceed the number of bits in the data frame.

Figure 3–55. Address Match Reception



The SMAT register has a bank address of 7h, and a port address of 2h.

Summary of address match protocol:

1) You must enable the asynchronous serial mode.
2) Parity and number of data bits are selected.
3) The serial port address (up to 9 bits) is written into the SMAT register.
4) Address match protocol is enabled by setting RINTQ2 (bit 10) of SCON to 1; this also puts serial port into sleep mode.
5) Reception is disabled.
6) Data is shifted into the RSR register and compared against the contents of the SMAT register.
7) If a match does not occur, the data in the RSR is discarded, and the serial port remains in sleep mode.
8) If a match does occur, the serial port wakes up and generates an interrupt (RXINT) to the CPU.
9) Reset the RINTQ2 bit of SCON to 0 to enable normal reception.
10) Data is clocked into RSR, generating an interrupt RXINT to the CPU every time a complete word is received.
11) You determine that the complete message has been received and puts the serial port back into sleep mode by setting RINTQ2 of SCON back to 1.
12) The serial port waits for another address match.

3.13 Serial Port (TMS320C17)

Two of the I/O ports on the TMS320C17 are dedicated to the serial port and companding hardware. I/O port 0 is dedicated to control register 0, which controls the serial port, interrupts, and companding hardware. I/O port 1 accesses control register 1, as well as both serial port channels and the companding hardware. The six remaining I/O ports are available for external parallel interfaces.

The on-chip dual-channel serial port provided on the TMS320C17 is capable of full-duplex serial communications and direct interface to combo-codec PCM systems, serial A/D converters, and other serial systems. The interface signals are directly compatible with codecs and many other serial devices and require a minimum of external hardware. For additional information on combo-codecs, refer to the *TCM29C13/C14/C16/C17 Combined Single-Chip PCM Codec and Filter* data sheet.

Two receive and two transmit registers are mapped into I/O port 1 and operate with 8-bit data samples. Either internal or external framing signals for serial data transfers (MSB first) are selected via the system control register. The serial port clock, SCLK, provides the bit timing for transfers with the serial port and may be either an input or an output. A framing pulse signal supplies framing pulses for combo-codec circuits, a sample clock for voice-band systems, or a timer for control applications. The serial port is accessed through IN and OUT instructions. A block diagram of the serial port and companding hardware is shown in Figure 3–56.

3.13.1 Receive Registers

Two receive registers are mapped into I/O port 1 via the port decode logic. Data is clocked into the shift registers on the next eight negative serial clock (SCLK) transitions after an active framing pulse is detected. SCLK controls the bit-level timing for all serial port data transfers. Note that the MSB is always shifted first.

On an active framing pulse, serial data is clocked into the receive registers from the DR pins. Channel 0 data is received in shift register RS0 from pin DR0, and channel 1 data is received in shift register RS1 from pin DR1. To read the data from the registers, an IN instruction is executed from port 1. On the first IN instruction after a framing pulse, channel 0 data is output onto the external data bus where it is read by the CPU. On the second IN instruction, channel 1 data is output onto the external data bus.

An active framing pulse initiates the receive operation, as shown in Figure 3–57. External framing pulses ($\overline{FSR}$) are active low, and the internal framing (FR) signal is active high. With external framing ($\overline{FSR}$), the falling edge of the framing pulse gates the serial port clock to the receive shift registers, and the data is clocked into the shift registers on the next eight consecutive negative transitions of the clock.

Figure 3–56. Serial Port and Companding Hardware

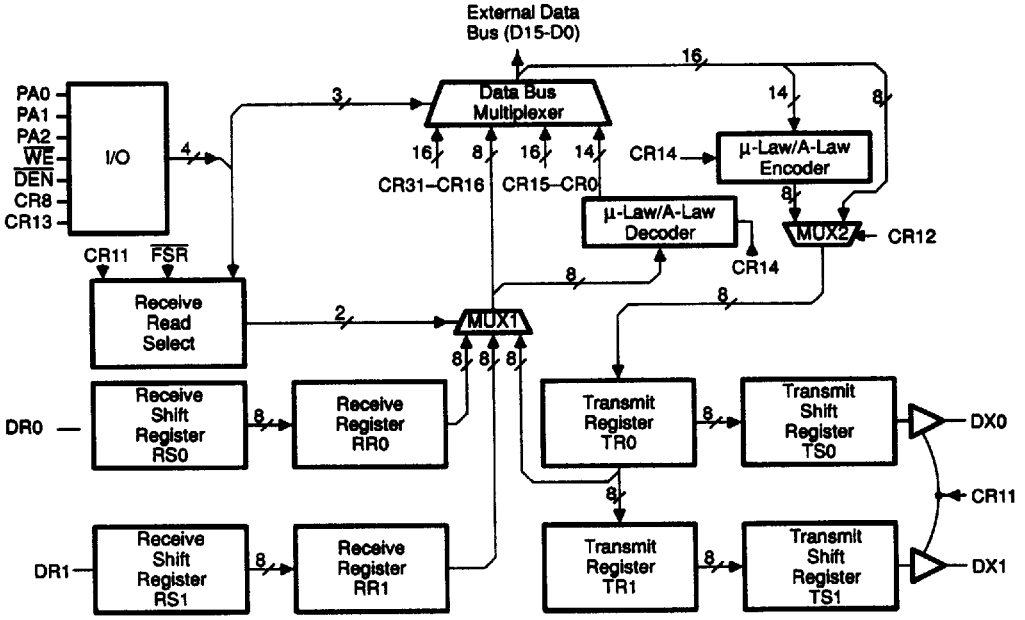
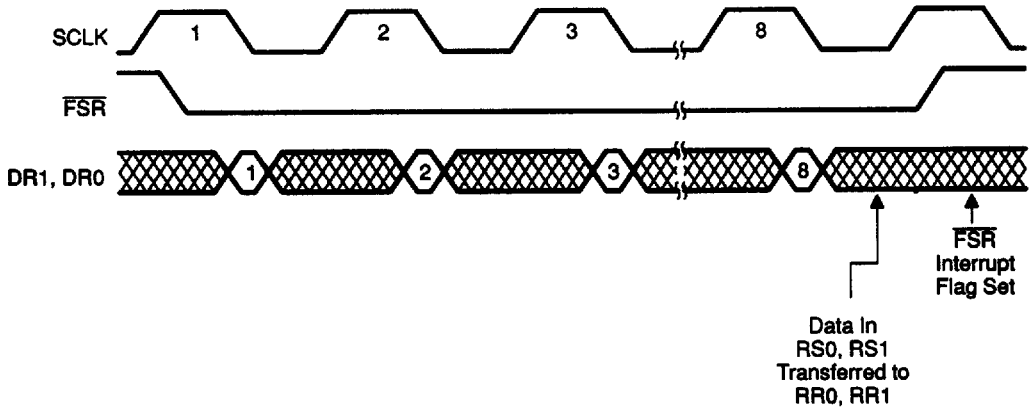


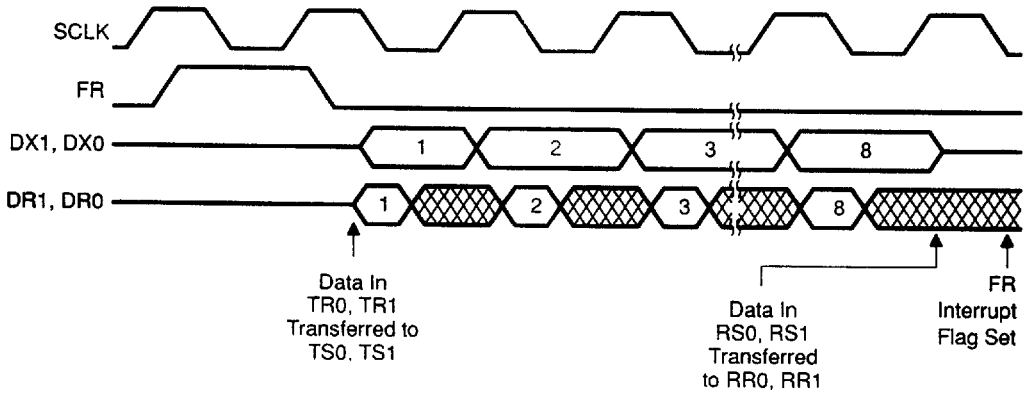
Figure 3–57. Receive Timing for External Framing



The rising edge of the framing pulse transfers the data from the receive shift registers to the receive registers and sets the $\overline{FSR}$ flag bit (CR1) in the system control register (see Figure 3–57), causing an interrupt to occur if the $\overline{FSR}$ is enabled. External framing pulses are sensed during the high portion of the SCLK cycle and latched internally with the falling edge of SCLK. Only one $\overline{FSR}$ state can be detected per SCLK period.

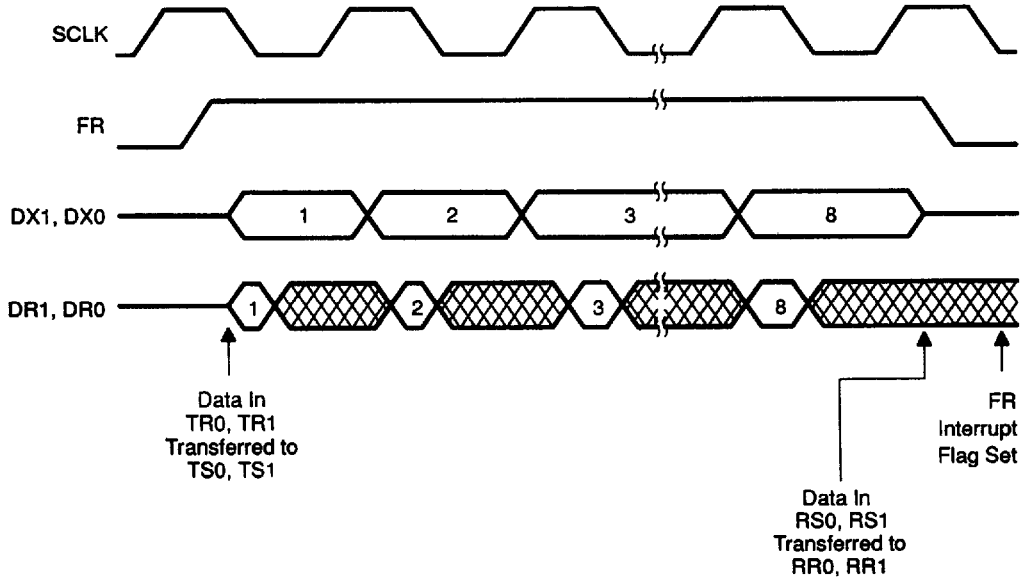
Internal framing (FR) pulses can be selected in either fixed data-rate or variable data-rate modes for combo-codec interface. With the fixed data-rate mode, the FR pulse is one SCLK cycle wide, and appears in the cycle preceding the first data bit. The falling edge of the pulse initiates both the transmit and receive operations, as shown in Figure 3–58. Received data is clocked into the receive shift registers on the next eight consecutive negative transitions of the clock. After data bit 8 has been received, data is transferred from the receive shift registers to the receive registers, and an interrupt is generated when the FR flag bit (CR3) is set in the system control register, thus causing an interrupt to occur if enabled.

Figure 3–58. Fixed Data-Rate for Internal Framing



In the variable data-rate mode shown Figure 3–59, the FR pulse is eight SCLK cycles wide and appears in the same SCLK cycle as the first data bit. The rising edge of the pulse initiates the transmit and receive operations. The falling edge of the pulse transfers data from the receive shift registers to the receive registers and sets the FR flag bit (CR3) in the system control register, causing an interrupt to occur if enabled

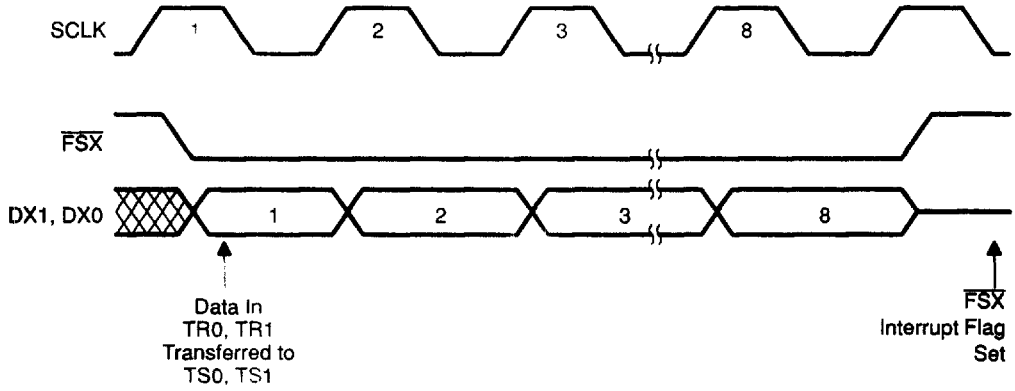
Figure 3–59. Variable Data-Rate for Internal Framing



3.13.2 Transmit Registers

Two transmit registers are mapped into I/O port 1 via the port decode logic. The transmit registers are connected to the port 1 data bus in a FIFO (first in, first out) configuration. On the first OUT instruction to port 1 after a framing pulse, the data to be transmitted is put into transmit register TR0. On the next framing pulse, the TR0 contents are latched into transmit shift register TS0, and the data is transmitted on channel 0 (pin DX0) on the next eight positive transitions of the serial port clock (SCLK), as shown in Figure 3–60. External framing pulses ($\overline{FSX}$) are active low, and the internal framing (FR) signal is active high. Data sent to port 1 is always put into the transmit registers. Only when control register bit 11 (CR11) is high, is the data enabled onto the transmit pins. The transmit pins are in the high-impedance state when not transmitting. External framing pulses are sensed during the high portion of the SCLK cycle and latched internally with the falling edge of SCLK. Only one FSX state can be detected per SCLK period.

Figure 3–60. Transmit Timing for External Framing



Internal framing (FR) pulses can be selected in either fixed data-rate or variable data-rate modes for combo-codec interface. In the fixed data-rate mode, the FR pulse is one SCLK cycle wide and appears in the cycle preceding the first data bit. The falling edge of the pulse initiates both the transmit and receive operations, as shown in Figure 3–60. Data is transferred from the transmit registers to the transmit shift registers. Transmitted data is clocked into the transmit shift registers on the next eight consecutive negative transitions from the clock. After data bit 8 has been transmitted, an interrupt is generated when the FR flag bit (CR3) is set in the system control register, thus causing an interrupt to occur if enabled.

In the variable data-rate mode shown in Figure 3–59, the FR pulse is eight SCLK cycles wide and appears in the same SCLK cycle as the first data bit. The rising edge of the pulse initiates the transmit and receive operations. The falling edge of the pulse sets the FR flag bit (CR3) in the system control register, causing an interrupt to occur if enabled.

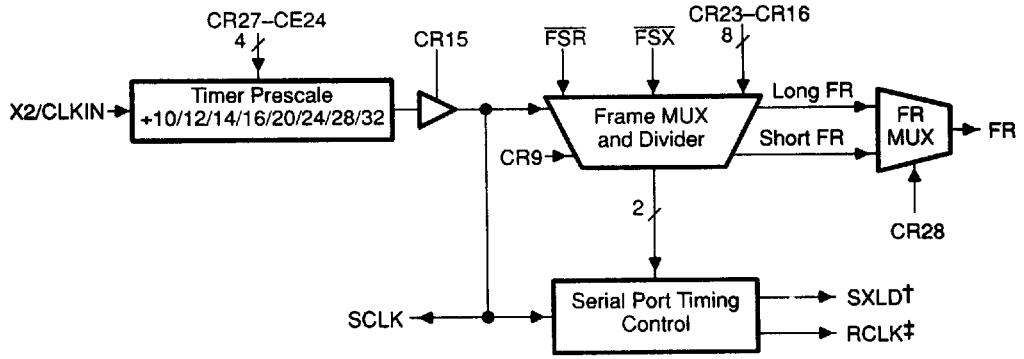
When two OUT instructions to port 1 are executed between framing pulses, both transmit registers are loaded with data for transmission. The first OUT instruction loads data into transmit register TR0. The second OUT pushes the data from TR0 into TR1 and puts the new data into TR0. On an active framing pulse edge, the transmit register contents are latched into the transmit shift registers, and the data clocked out on the next eight consecutive positive transitions of SCLK. Thus, for single-channel operation, only one OUT instruction to port 1 should be executed between framing pulses to ensure data transmission on channel 0. Only TR0 may be read back to the serial port data bus by an IN instruction. This feature is used for the parallel companding mode.

Both transmit channels always output data on an active framing pulse when CR11 is high. During single-channel operation (using channel 0), channel 1 still transmits the data from transmit register TR1. Transmit channel 1 cannot be disabled during single-channel operation.

3.13.3 Timing and Framing Control

The serial port timing and framing control is shown in Figure 3–61. The serial port clock (SCLK) provides the timing control for data transfers with the serial port. SCLK may be configured as either an input or an output through the control register. As an input, SCLK is an external serial system clock that provides the framing synchronization and timing for the serial port. As an output, SCLK provides the system clock for standalone serial applications and is derived from the microcomputer system clock (X2/CLKIN).

Figure 3–61. Serial Port Timing and Framing Control



† SXLD = Load transmit shift registers (TS0,TS1) from transmit registers (TR0,TR1)
‡ RCLK = Load receive registers (RR0,RR1) from receive shift registers (RS0,RS1).

The serial-port clock prescaler determines the divide ratio for SCLK when configured as an output. The TMS320C17 system clock (X2/CLKIN) is input to the prescaler, along with control register bits CR27–CR24. Table 3–26 shows the prescale divide ratios selectable as divide by 10, 12, 14, 16, 20, 24, 28, and 32 through system control register bits CR27–CR24. These divide ratios are available only for SCLK when it is configured as an output from the device (see Section 3.12 for control register bit configurations).

The frame multiplexer determines which framing pulses cause serial-port data transfers to occur and configures the internal framing pulse (FR) frequency. The inputs to the multiplexer are SCLK, control register bit 9 (CR9), control register bits CR23–CR16, external transmit framing ($\overline{FSX}$) pulse, and external receive framing ($\overline{FSR}$) pulse. The outputs of the multiplexer go to the serial-port control for receive and transmit timing generation for the serial-port registers and to the FR multiplexer for determining which FR framing pulse is generated.

The outputs of the frame counter are input to the FR multiplexer for selection of long or short FR pulses. The short FR pulse provides fixed data-rate framing pulses for standalone serial interface to the Texas Instruments TCM29Cxx family of combo-codec circuits. The long FR framing pulse provides variable data-rate framing pulses to the combo-codec.

The FR frequency is determined at the beginning of the framing pulse cycle. The FR frequency is equal to SCLK/(CNT + 2) where CNT is the binary value of CR23–CR16. When you are reconfiguring the frequency, the upper control register bits determine the new divide ratio. However, the new frequency is not implemented until the next FR framing pulse.

Table 3–26. Serial Clock (SCLK) Divide Ratios (X2/CLKIN = 20.48 MHz)

CR27	CR26	CR25	CR24	Divide Ratio	SCLK Freq. MHz
0	0	0	0	32	0.640
0	0	0	1	28	0.731
0	0	1	0	24	0.853
0	1	0	0	20	1.024
1	0	0	0	16	1.280
1	0	0	1	14	1.463
1	0	1	0	12	1.706
1	1	0	0	10	2.048

3.14 Companding Hardware (TMS320C17)

The on-chip companding hardware enables the TMS320C17 to compand (COMpress and exPAND) data in either μ-law or A-law format with either sign-magnitude or 2s-complement numbers. The companding standard employed in the United States and Japan is μ-law. The European companding standard is referred to as A-law. Configuration and connections of the encoder and decoder (see Figure 3–56) are controlled through the system control register.

When sign magnitude is selected, the μ-law encoding and decoding require a bias adjustment in the sample value. For μ-law encoding, a bias of 33 must be added to the sign magnitude before encoding; likewise, after μ-law decoding, the bias of 33 must be subtracted from the sign-magnitude value. No additional bias adjustment is required for μ-law encoding and decoding when the selected conversion uses two's-complement notation. Note that A-law encoding and decoding do not require a bias adjustment in either case.

Upon reset, theTMS320C17 is programmed to operate in sign-magnitude mode. This mode can be changed by modifying control register bit 29 (CR29). Refer to the *TCM29C13/TCM29C14/TCM29C16/TCM29C17 Combined Single-Chip PCM Codec and Filter* data sheet for further information on companding. If you want software companding without the use of companding hardware, descriptive algorithms are given in the book, *Digital Signal Processing Applications with the TMS320 Family* (literature number SPRA012A); refer to the application report *Companding Routines for the TMS32010/TMS32020.*

The specification for μ-law and A-law log PCM is part of the CCITT G.711 recommendation. Part of the coding format specifies certain bits to be inverted before transmission or upon receipt of transmitted data. The companding hardware in the TMS320C17 implements the bit inversion as well as the logarithmic compression and decompression. For the μ-law format, all of the data bits are inverted. Refer to the data sheet in Appendix A for diagrams of the codec interface circuits used for μ-law and A-law formats on the TMS320C17 devices.

Data may be companded via four modes: serial-port encode, serial-port decode, parallel encode, and parallel decode. In the serial mode, transmitted data is encoded according to the specified companding law, and received data is decoded to either sign-magnitude or 2s-complement format. In the parallel modes, encoding or decoding is performed on data from the RAM for computations within the device. Note that in parallel mode when 2s-complement notation is selected, at least one instruction must be inserted between successive OUT and IN instructions to I/O port 1.

Table 3–27 shows the control register bit combinations that determine the serial or parallel modes of the companding hardware operation. Note that the serial and parallel companding modes require separate control register settings. When you are using the serial mode, parallel companding is not available unless the control register is reconfigured.

Table 3–27. Serial- and Parallel-Mode Bit Configurations

CR Bit #			Mode of Operation
13	12	11	
0	0	0	Parallel mode. Encoder and decoder are disabled. No operation performed on data written to or read from port 1.
0	0	1	Serial mode. Encoder and decoder are disabled.The transmit registers are enabled for data transmission on an active framing pulse. The 8-bit value written to port 1 is transmitted, and the 8-bit value in the receive register is read with an IN instruction from port 1.
0	1	0	Parallel encode. Encoder is enabled. A linear sample written to port 1 with an OUT instruction is compressed to 8-bit log PCM. The 8-bit value is then read from port 1 with an IN instruction.
0	1	1	Serial encode. Encoder is enabled. A linear sample written to port 1 is compressed to 8-bit log PCM and put into the transmit register for transmission on an active framing pulse.
1	0	0	Parallel decode. Decoder is enabled. An 8-bit log PCM data written to port 1 is decoded to linear notation with an IN instruction from port 1.
1	0	1	Serial decode. Decoder is enabled. An 8-bit log PCM sample from one of the receive registers is expanded to linear notation with an IN instruction from port 1.
1	1	0	Parallel encode and decode. Encoder and decoder enabled. In this state, data is compressed on an OUT instruction to port 1 and then expanded with the IN instruction from the port.
1	1	1	Serial encode and decode. Encoder and decoder enabled. Linear data written to port 1 is encoded and put into one of the transmit registers for serial transmission. The 8-bit log PCM data from one of the receive registers is decoded with an IN instruction from port 1.

3.14.1 μ-Law/A-Law Encoder

The encoder compresses linear PCM (14 bits of dynamic range for μ-law format or 13 bits of dynamic range for A-law format) to 8-bit logarithmic PCM. Selection between μ-law or A-law conversion is determined by the system control register bit 14(CR14). This bit is input directly to the encoder to determine the conversion law to be used. The μ-law conversion is performed if CR14 is logic 0, and A-law conversion is performed if CR14 is logic 1. Data is input to the encoder from the data bus with an OUT instruction to port 1. The converted 8-bit log PCM sample is then presented to the multiplexer (MUX2 shown in

Figure 3–56). The multiplexer controls whether the encoder output or the eight low-order data bus bits are input to transmit register TR0 of the serial port. Note that the transmit registers are connected to the port 1 data bus in a FIFO (first in, first out) configuration. The encoder compresses data written to port 1 at all times, but the output is enabled to the TR0 only when CR12 is logic 1.

In the serial encode mode, data written to port 1 is encoded, and the value is put into transmit register TR0. The transmit register is then loaded with the 8-bit value on an active framing pulse, and the 8 bits are clocked out on the positive edge of SCLK.

For the parallel encode mode, the linear-PCM value is written to port 1 with an OUT instruction. The encoded 8-bit value is then stored in TR0. An IN instruction from port 1 reads TR0 to the data bus for storage in RAM. Only one OUT and one IN instruction to port 1 for each data sample is allowed in the parallel-encode mode. If there are two OUT instructions to port 1, the first sample is pushed into transmit register TR1, which cannot be read back to the data bus. Note that when 2s-complement notation is selected, there must be at least one instruction executed after the OUT instruction to port 1 and before the IN instruction from port 1.

3.14.2 μ-Law/A-Law Decoder

The μ-law/A-law decoder converts 8-bit log-PCM samples to linear PCM. The conversion law selection is governed by control register bit 14 (CR14). The μ-law conversion is performed if CR14 is logic 0, and A-law conversion if CR14 is logic 1. Data input to the decoder may come from either the serial port receive registers or transmit register TR0. The multiplexer (MUX1 shown in Figure 3–56) sends data to the data bus either through the decoder or directly to the bus. This multiplexer is controlled in part by control register bit 13 (CR13). If this bit is logic 0, the multiplexer output is sent to the data bus directly. If the bit is logic 1, the multiplexer output is sent to the data bus through the decoder.

In the serial decode mode, received data from the serial-port receive registers is input to the decoder from the multiplexer, and the received data is decoded according to either μ-law or A-law format.

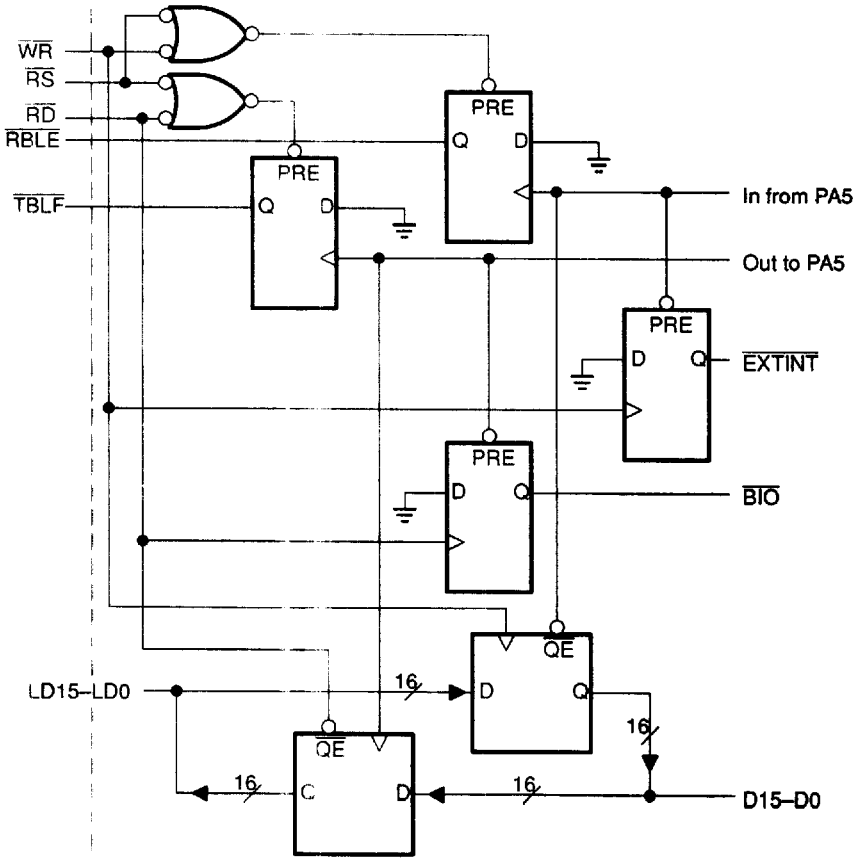
For the parallel decode mode, the 8-bit PCM sample to be decoded is written to port 1 with an OUT instruction. This stores the sample in transmit register TR0. The sample is then decoded by reading the value from port 1 with an IN instruction. The IN instruction brings the sample from TR0 through the multiplexer (MUX1) to the decoder, which performs the expansion on the 8-bit sample. Again, there should be only one OUT and one IN instruction to port 1 for each sample to be decoded and to avoid losing a sample in transmit register TR1. Note that when 2s-complement notation is selected, there must be at least one instruction must be executed after the OUT instruction to port 1 and before the IN instruction from port 1.

3.15 Coprocessor Port (TMS320C17)

The coprocessor port on the TMS320C17 provides a direct interface to most 4-/8-bit microcomputers and 16-/32-bit microprocessors. The port is accessed through I/O port 5 by using IN and OUT instructions. The coprocessor interface allows the device to act as a peripheral (slave) microcomputer to a microprocessor, or as a master to a peripheral microcomputer such as the TMS7042. The coprocessor port is enabled by setting MC/$\overline{PM}$ and MC low. The microcomputer mode is enabled by setting these two pins high. (Note that the MC/$\overline{PM}$ and MC pins must be in the same state.) In the microcomputer mode, the 16 data lines are used for the 6 parallel 16-bit I/O ports.

Interprocessor communication through the coprocessor interface (see Figure 3–62) is accomplished asynchronously as in memory-mapped I/O operations. In coprocessor mode, the 16-bit data bus is reconfigured to operate as a 16-bit latched bus interface. Control bit 30 (CR30) in control register 1 configures the coprocessor port to either an 8-bit or a 16-bit length for data transfers. Use of the HI/$\overline{LO}$ pin allows the full 16-bit data latches to be used even when the 8-bit mode is selected.

Figure 3–62. TMS320C17 Simplified Coprocessor Port Logic Diagram



Several key characteristics of the coprocessor interface are listed below.

- The $\overline{BIO}$ and $\overline{EXINT}$ signals are internal to the processor. No inputs should be made on the $\overline{BIO}$ and $\overline{EXINT}$ pins.
- Only transfers made when HI/$\overline{LO}$ is in a low state can activate the internal $\overline{BIO}$ and $\overline{EXINT}$ signals.
- The interrupt condition is kept internally until it is cleared by the TMS320C17 reading the data in the coprocessor port latch. The interrupt flag cannot be cleared until the port is read. Four instruction cycles must take place between the read (IN instruction) from the coprocessor port (PA5) and the write to control register CR0 to clear the interrupt flag.
- When the TMS320C17 reads the coprocessor port, it clears the data in the latch.
- The16 data lines (LD15–LD0) remain in a high-impedance state unless a logic low is asserted on $\overline{RD}$. When $\overline{RD}$ is asserted, the TMS320C17 drives the data bus with the data in the coprocessor port latch.

The following sequences of events occur, depending upon the configuration and use of the coprocessor port:

- 16-bit data interface (CR30 = 1):
- All 16 bits of the data port are available for 16-bit transfers to16-/32-bit microprocessors.
- The HI/$\overline{LO}$ pin is maintained at a logic low level for all transfers.
- Transfers to the TMS320C17 (see Figure 3–63):
  1) The $\overline{WR}$ signal is driven low by the microprocessor.
  2) The $\overline{RBLE}$ (receive buffer latch empty) signal transitions to a logic high level in response to $\overline{WR}$.
  3) Data is written from LD15–LD0 to the receive buffer latch when the $\overline{WR}$ signal is driven high by the microprocessor.
  4) The internal $\overline{EXINT}$ signal is generated, causing the interrupt flag to be set in the TMS320C17
  5) The TMS320C17 responds to the interrupt condition and reads port 5 by using an IN instruction.
  6) The receive buffer is cleared. (Subsequent reads by the TMS320C17 are zero value.)
  7) The $\overline{RBLE}$ signal transitions to a logic low level, signaling the microprocessor that the receive buffer is empty.
  8) The internal $\overline{EXINT}$ signal is removed, allowing the interrupt flag to be cleared.
  9) The interrupt flag is cleared by writing to control register 0.

Transfers from the TMS320C17 (see Figure 3–64):

1) The $\overline{RD}$ signal is driven low by the microprocessor.
2) The TBLF (transmit buffer latch full) signal transitions to a logic high level in response to $\overline{RD}$.
3) Data is driven from the transmit buffer latch to LD15–LD0 until the $\overline{RD}$ signal is driven high by the microprocessor.
4) The internal $\overline{BIO}$ signal transitions to a logic low level, indicating to the TMS320C17 that the transmit buffer is empty.
5) The TMS320C17 responds to the $\overline{BIO}$ condition and writes to port 5 by using an OUT instruction.
6) The $\overline{TBLF}$ signal transitions to a logic low level, signaling the microprocessor that the transmit buffer is full.
7) The internal $\overline{BIO}$ signal transitions back to a logic high state.

- 8-bit data interface (CR30 = 0):

Only the least significant eight bits of the data port are available for 8-bit transfers to 4-/8-bit microcomputers.

Eight-bit microcomputers may complete full 16-bit transfers by transferring data firstwith the HI/$\overline{LO}$ signal in a logic high state (steps 1 through 4 below) and then with HI/LO in a logic low state. Composing 16-bit data in this manner requires two external bus cycles but only one internal port access. The HI/$\overline{LO}$ pin may be maintained at a logic low level if only 8-bit transfers are desired.

Transfers to the TMS320C17 (see Figure 3–63):

1) The HI/$\overline{LO}$ signal is driven high by the microcomputer to allow transfers to the upper eight bits of the internal latch.
2) The $\overline{WR}$ signal is driven low by the microcomputer.
3) The $\overline{RBLE}$ (receive buffer latch empty) signal transitions to a logic high level.
4) Data is written from LD7–LD0 to the receive buffer latch (D15–D8) when the $\overline{WR}$ signal is driven high by the microcomputer.
5) The HI/$\overline{LO}$ signal is driven low by the microcomputer to allow transfers to the lower eight bits of the internal latch.
6) The $\overline{WR}$ signal is driven low by the microcomputer.
7) Data is written from LD7–LD0 to the receive buffer latch (D7–D0) when the $\overline{WR}$ signal is driven high by the microcomputer.
8) The internal $\overline{EXINT}$ signal is generated, causing the interrupt flag to be set in the TMS320C17

9) The TMS320C17 responds to the interrupt condition and reads port 5 by using an IN instruction.

10) The receive buffer is cleared. (Subsequent reads by the TMS320C17 will be zero value.)

11) The $\overline{\text{RBLE}}$ signal transitions to a logic low level, signaling the microcomputer that the receive buffer is empty.

12) The internal $\overline{\text{EXINT}}$ signal is removed, allowing the interrupt flag to be cleared.

13) The interrupt flag is cleared by writing to control register 0.

Transfers from the TMS320C17 (see Figure 3–64):

1) The HI/$\overline{\text{LO}}$ signal is driven high by the microcomputer to allow transfers from the upper eight bits of the internal latch.

2) The $\overline{\text{RD}}$ signal is driven low by the microcomputer.

3) Data is driven from the transmit buffer latch (D15–D8) to LD7–LD0 until the $\overline{\text{RD}}$ signal is driven high by the microcomputer.

4) The HI/$\overline{\text{LO}}$ signal is driven low by the microcomputer to allow transfers from the lower eight bits of the internal latch.

5) The $\overline{\text{RD}}$ signal is driven low by the microcomputer.

6) The TBLF (transmit buffer latch full) signal transitions to a logic high level.

7) Data is driven from the transmit buffer latch (D7–D0) to LD7–LD0 until the $\overline{\text{RD}}$ signal is driven high by the microcomputer.

8) The internal $\overline{\text{BIO}}$ signal transitions to a logic low level, indicating to the TMS320C17 that the transmit buffer is empty.

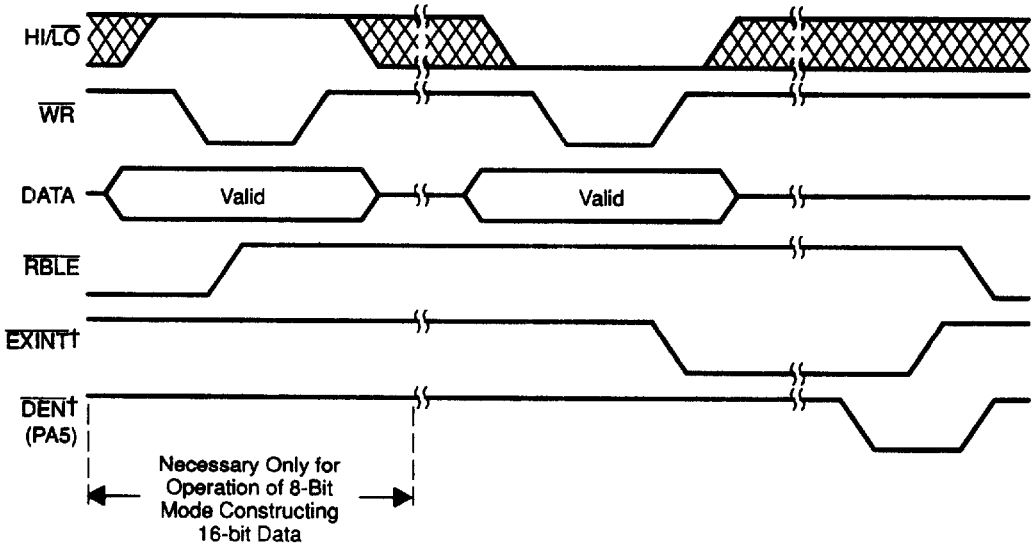
9) The TMS320C17 responds to the $\overline{\text{BIO}}$ condition and writes to port 5 using an OUT instruction.

10) The TBLF signal transitions to a logic low level, signaling the microcomputer that the transmit buffer is full.

11) The internal $\overline{\text{BIO}}$ signal transitions back to a logic high state.

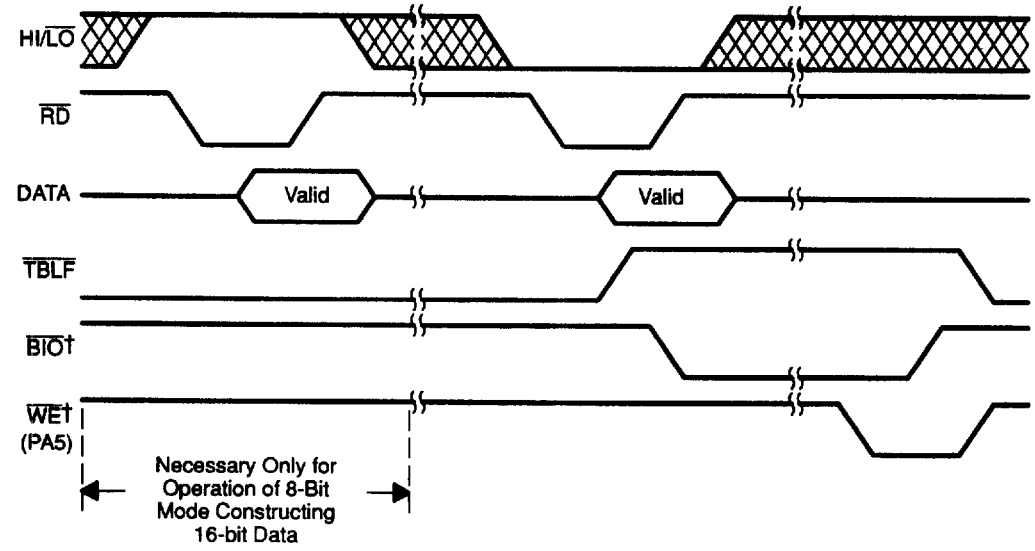
Examples of the use of a coprocessor interface are provided in subsection 6.2.3 and in the data sheet in Appendix A.

Figure 3–63. External Write Timing to the Coprocessor Port



† Internal signals

Figure 3–64. External Read Timing From the Coprocessor Port

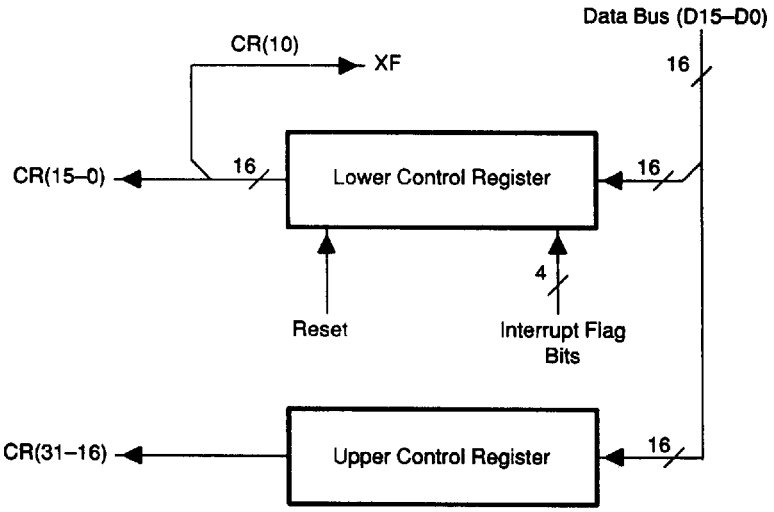


† Internal signals

3.16 System Control Register (TMS320C17)

The TMS320C17 features hardware that supports serial applications. This hardware is interfaced to the microcomputer portion of the device via the external data bus (D15–D0) and is controlled by a 32-bit system control register (see Figure 3–65). This makes additions to the TMS320 instruction set unnecessary.

Figure 3–65. System Control Register



The lower 16 register bits (CR15–CR0) are accessed through port 0. These bits control interrupts, serial port configuration, the external logic output flag, internal and external framing pulses, and the μ-law/A-law encoder and decoder. The interrupt inputs ($\overline{EXINT}$, $\overline{FSX}$, $\overline{FSR}$, and FR) are synchronized to CLKOUT and control the interrupt flag bits (CR3–CR0). The interrupts are maskable via the interrupt enable bits (CR7–CR4). Bit 8 (CR8) controls I/O port 1 configuration.

The upper 16 bits (CR31–CR16) are accessed through port 1. These bits control the internal framing pulse (FR) output frequency, serial clock divide ratios, pulse-width control for the FR framing pulse, and companding conversions. The bit width of the coprocessor mode is controlled by CR30.

The external data bus provides on-chip communication with the system control register, serial port, companding hardware, and coprocessor port. With a write to port 0, the lower control register is addressed and data latched into the register by the rising edge of the write enable ($\overline{WE}$) signal. To write to the upper control register bits, bit 8 of the lower control register must be set to logic 1. If CR8 is logic 0, a write to port 1 accesses the serial port and companding hardware.

Table 3–28 gives a detailed description of the control register bits and their operation. The control register bits are configured through OUT instructions to

port 0 and port 1. $\overline{WE}$ goes low during the first cycle of the OUT instruction, enabling the port data onto the external data bus. The control register bits are latched on the rising edge of $\overline{WE}$. These bits require a propagation delay to access the appropriate hardware (see Appendix A for timing information). To avoid receiving an external framing pulse at this time, this write delay should be considered during reconfiguration of (writing to) the control register. If an external framing pulse is received while the control register is being reconfigured, the pulse may not be detec ted, and the serial port registers may contain random data (see Section 3.13 for further details).

Table 3–28. Control Register Bit Definitions

CR Bit#	Description
3–0	Interrupt flags. When an interrupt occurs on any of the four maskable interrupts, the appropriate flag is set to logic 1, whether the interrupt is enabled or disabled. To clear the flag, a logic 1 is written to the appropriate bit by an OUT instruction to port 0. The bits may be read by an IN instruction to determine interrupt sources when multiple interrupts are enabled. Bit # Flag 0 $\overline{EXINT}$ 1 $\overline{FSR}$ 2 $\overline{FSX}$ 3 FR
7–4	Interrupt enable bits. When one of these bits is set to logic 1, an interrupt occurring on that input sets the appropriate flag and activates the microcomputer interrupt circuitry. When disabled, the interrupt flag is still set, but the device is not interrupted. Bit # Flag 4 $\overline{EXINT}$ 5 $\overline{FSR}$ 6 $\overline{FSX}$ 7 FR
8	Port 1 control bit. When this bit is set to logic 0, I/O port 1 is connected to either the serial port registers or the companding hardware, depending on the state of CR11. When the bit is set to logic 1, I/O port 1 is connected to the upper control register. This bit must be set with an OUT instruction to port 0 before port 1 may access the upper control register bits CR31–CR16.
9	External framing enable. This bit controls which framing pulses cause serial port data transmission to occur. When the bit is set to logic 0, serial port transmit and receive operations occur simultaneously and are controlled by the internal framing (FR) pulse. When the bit is set to logic 1, transmit operations are controlled by the external transmit framing ($\overline{FSX}$) pulse, and receive operations are controlled by the external receive framing ($\overline{FSR}$) pulse

Table 3–28. Control Register Bit Definitions (Continued)

CR Bit#	Description
10	XF output latch. This bit controls the logic level of the external logic output flag (XF) pin. A write delay time occurs during reconfiguration of this latch (see Appendix A for timing information).
11	Serial port enable. When this bit is set to logic 0, the transmit and receive registers are disabled so that the parallel companding mode can be used. When the bit is set to logic 1, the serial port registers are enabled and data transfers with the serial port are via OUT and IN instructions to port 1. A reset sets this bit to zero.
12	μ-law/A-law encoder enable. When set to logic 0, the encoder is disabled. When set to logic 1, the encoder is enabled, and data written to port 1 is μ-law or A-law encoded. The encoder must be enabled for compression of linear data in both the serial and parallel modes of operation.
13	μ-law/A-law decoder enable. When this bit is set to logic 0, the decoder is disabled. When set to logic 1, the decoder is enabled, and data read from port 1 is μ-law or A-law decoded to linear format. The decoder must be enabled for expansion of log PCM data in both the serial and parallel modes of operation.
14	μ-law/A-law encode/decode select. When this bit is set to logic 0, the companding hardware performs μ-law conversion. When this bit is set to logic 1, the companding hardware performs A-law conversion.
15	Serial clock control. When this bit is set to logic 0, the serial port clock (SCLK) is an output, and its frequency is derived from the microcomputer system clock, X2/CLKIN. When the bit is set to logic 1, SCLK is an input that provides the clock for all data transfers with the serial port and the frame counter in timing logic. A reset sets this bit to one.
23–16	Frame counter modulus. The value of these bits determines the divide ratio for the FR output frequency. The FR frequency is given as SCLK/(CNT + 2) where CNT is a binary value of CR23–CR16. The following should be noted when configuring the divide ratio: 1. CNT must be in the range given by $7 \leq CNT \leq 254$. 2. Bits are operational whether SCLK is an input or an output.
27–24	SCLK prescale control bits. As an output, SCLK is derived from the microcomputer system clock, X2/CLKIN. Prescale divide ratios are selectable through these control bits (see subsection 3.9.3 for the available divide ratios).
28	FR pulse-width control. This bit controls the pulse width of the FR output to select data-transfer rates with combo-codec circuits. When this bit is set to logic 0, the FR output framing pulse is one SCLK cycle wide for the fixed data-rate mode and appears in the serial-clock cycle preceding the first serial-bit transmission. When set to logic 1, the FR output framing pulse is eight SCLK cycles wide for the variable data-rate mode. In this mode, the framing pulse is active high for the duration of the eight bits transmitted and received.

Table 3–28. Control Register Bit Definitions (Continued)

CR Bit#	Description
29	Twos-complement μ-law/A-law conversion enable. When this bit is set to logic 0, sign-magnitude companding is enabled. When the bit is set to logic 1, 2s-complement companding is enabled. When 2s-complement companding has been selected along with the parallel companding mode of operation, one instruction must be inserted between successive OUT and IN instructions to port 1. A reset sets this bit to zero.
30	8-/16-bit length coprocessor mode select. When this bit is set to logic 0, the 8-bit byte length is used. When the bit is set to logic 1, the16-bit word length is selected.
31	Reserved for future expansion. This bit should be set to zero.

Chapter 4

Assembly Language Instructions

The instruction set of the TMS320C1x generation processors supports numeric-intensive signal processing operations and general-purpose applications, such as high-speed control. The instruction set shown in Table 4–2 consists primarily of single-cycle, single-word instructions, facilitating execution rates of up to 6.25 million instructions per second. Only infrequently used branch and I/O instructions are multicycle

A single-cycle instruction (MPY) executes multiplication operations. For ease of use in a Harvard architecture, table read (TBLR) and table write (TBLW) instructions transfer information between data and program memory. The IN and OUT instructions permit a data word to be read into the on-chip RAM in only two cycles. The SUBC (conditional subtract) instruction performs the shifting and conditional branching necessary to implement a divide efficiently and quickly.

Note:

Throughout this book, 'C14 designates all devices with a 14 suffix (C14/E14/P14), 'C15 all devices with a 15 suffix (C15/E15/LC15/P15), and 'C17 all devices with a 17 suffix (C17/E17/LC17/P17), unless otherwise noted.

This chapter describes the TMS320C1x assembly language instructions. Included in this chapter are the following major topics:

4.1 Memory Addressing Modes

The TMS320C1x instruction set provides three memory addressing modes:

- Direct
- Indirect
- Immediate

Both direct and indirect addressing can be used to access data memory. Direct addressing concatenates seven bits of the instruction word with the 1-bit data memory page pointer to form the 8-bit data memory address. Indirect addressing accesses data memory through the two auxillary registers. In immediate addressing, the data is based on a portion of the instruction word(s). The following sections describe each addressing mode and give the opcode formats and some examples for each mode.

4.1.1 Direct Addressing Mode

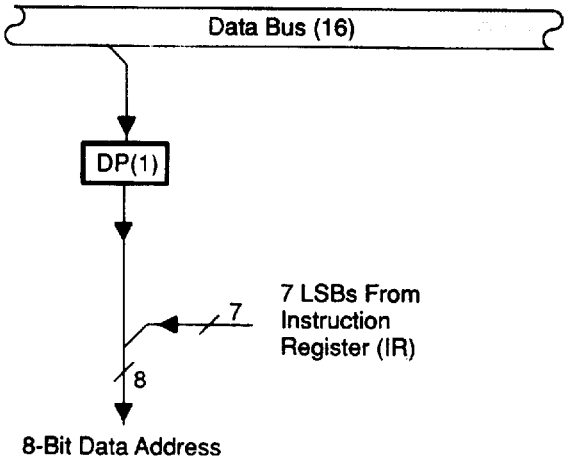
In the direct memory addressing mode, the instruction word contains the lower seven bits of the data memory address (dma). This field is concatenated with the one-bit data memory page pointer (DP) register to form the full 8-bit data memory address. This implements a paging scheme in which the first page contains 128 words and the second page contains 16/128 words. In a typical application, infrequently accessed system variables, such as those used when performing an interrupt routine, are stored on the second page. The 7-bit address in the instruction points to the specific location within that data memory page. The DP register is loaded through the LDP (load data memory page pointer), LDPK (load data memory page pointer immediate), or LST (load status bits from data memory) instructions. The data page pointer is part of the status register and thus can be stored in data memory.

Note:

The data page pointer is not initialized by reset and is therefore undefined after powerup. The TMS320C1x development tools, however, utilize default values for many parameters, including the data page pointer. Because of this, programs that do not explicitly initialize the data page pointer may execute improperly, depending on whether they are executed on a TMS320C1x device or with a development tool. Thus, it is critical that all programs initialize the data page pointer in software.

Figure 4–1 illustrates how the 8-bit data address is formed.

Figure 4–1. Direct Addressing Block Diagram



Direct addressing can be used with all instructions except CALL, the branch instructions, immediate operand instructions, and instructions with no operands. The direct addressing format is as follows:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Opcode								0	dma						

Bits 15 through 8 contain the opcode. Bit 7 = 0 defines the addressing mode as direct. Bits 6 through 0 contain the data memory address (dma), which can directly address up to 128 words (1 page) of data memory. Use of the data memory page pointer is required to address the full data memory space.

Example of direct addressing format:

ADD 9,5 Add to accumulator the contents of data memory location 9, left-shifted 5 bits.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	1	0	1	0	0	0	0	1	0	0	1

The opcode of the ADD 9,5 instruction is 05h and appears in bits 15 through 8. The notation *nn*h indicates that nn is a hexadecimal number. The shift count of 5h appears in bits 11 through 8 of the opcode. The data memory address 09h appears in bits 6 through 0

4.1.2 Indirect Addressing Mode

Indirect addressing forms the data memory address from the least significant eight bits of one of the two auxiliary registers, AR0 and AR1. This is sufficient to address all the data memory; no paging is necessary with indirect addressing. The auxiliary register pointer (ARP) selects the current auxiliary register. The auxiliary registers can be automatically incremented or decremented in parallel with the execution of any indirect instruction; this permits single-cycle manipulation of data tables. The increment/decrement occurs after the current instruction has completed executing.

In indirect addressing, the 8-bit addresses contained in the auxiliary registers may be loaded by the instructions LAR (load auxiliary register) and LARK (load auxiliary register immediate). The auxiliary registers may be modified by the MAR (modify auxiliary register) instruction or, equivalently, by the indirect addressing field of any instruction supporting indirect addressing. AR(ARP) denotes the auxiliary register selected by ARP.

The following symbols are used in indirect addressing:

- `*` Contents of AR(ARP) are used for data memory address.
- `*-` Contents of AR(ARP) are used for address, then decremented after data memory access.
- `*+` Contents of AR(ARP) are used for address, then increased after data memory access.

The indirect addressing format is as follows:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Opcode								1	0	INC	DEC	NAR	0	0	ARP

Note: NAR = new auxiliary register control bit.

Bits 15 through 8 contain the opcode, and bit 7 = 1 defines the addressing mode as indirect. Bits 6 through 0 contain the indirect addressing control bits.

Bit 3 and bit 0 control the ARP. If bit 3 = 0, the contents of bit 0 are loaded into the ARP after execution of the current instruction. If bit 3 = 1, the contents of the ARP remain unchanged. ARP = 0 defines the contents of AR0 as a memory address. ARP = 1 defines the contents of AR1 as a memory address.

Bit 5 and bit 4 control the auxiliary registers. If bit 5 = 1, the current auxiliary register is incremented by 1 after execution. If bit 4 = 1, the current auxiliary register is decremented by 1 after execution. If bit 5 and bit 4 are 0, then neither auxiliary register is increased or decreased. Bits 6, 2, and 1 are reserved and should always be programmed to 0.

The auxiliary registers may also be used for temporary storage via the load and store auxiliary register instructions, LAR and SAR, respectively.

The examples that follow illustrate the indirect addressing format. Indirect addressing is indicated by an asterisk (*) in these examples and in the TMS320C1x assembler.

Example 1:

ADD *+,8 Add to the accumulator the contents of the data memory address defined by the contents of the current auxiliary register. This data is left-shifted 8 bits before being added. The current auxiliary register is autoincremented by one. The opcode is 08A8h, as shown below.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	1	0	0	0	1	0	1	0	1	0	0	0

Example 2:

ADD *,8 The same as Example 1, but with no autoincrement; the opcode is 0888h.

Example 3:

ADD *-,8 The same as Example 1, except that the current auxiliary register is decreased by 1; the opcode is 0898h.

Example 4:

ADD *+,8,1 The same as Example 1, except that the auxiliary register pointer is loaded with the value 1 after execution; the opcode is 08A1h.

Example 5:

ADD *+,8,0 The same as Example 4, except that the auxiliary register pointer is loaded with the value 0 after execution; the opcode is 08A0h.

4.1.3 Immediate Addressing Mode

The TMS320C1x instruction set has five immediate operand instructions, in which the immediate operand is contained within the instruction word. These instructions execute within a single instruction cycle. The length of the constant operand is instruction-dependent. The immediate instructions are

LACK Load accumulator immediate short (8-bit constant)

LARK Load auxiliary register immediate short (8-bit constant)

LARP Load auxiliary register pointer (1-bit constant)

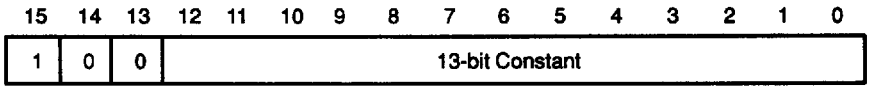
LDPK Load data memory page pointer immediate (1-bit constant)

MPYK Multiply immediate (13-bit constant)

The following examples illustrate immediate addressing format:

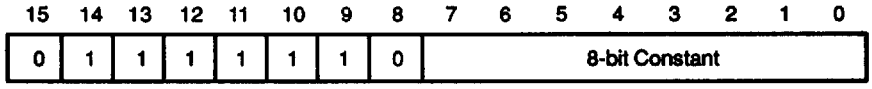
Example 1:

MPYK 2781 Multiply the value 2781 with the contents of the T register. The result is loaded into the P register.



Example 2:

LACK 221 Load the constant 221 into the lower eight bits of the accumulator, right-justified. The upper 24 bits of the accumulator are zero.



4.2 Instruction Set

The following sections list the symbols and abbreviations used in the TMS320C1x instruction set summary and in the instruction descriptions. The complete instruction set summary is organized according to function. The instructions are explained in detail in Section 4.3.

4.2.1 Symbols and Abbreviations

Table 4–1 lists symbols and abbreviations used in the instruction set summary on Table 4–2 and in the individual instruction descriptions.

Table 4–1. Instruction Symbols

Symbol	Meaning
A	Port address
ACC	Accumulator
ARn	Auxiliary Register n (AR0 and AR1 are predefined assembler symbols equal to 0 and 1, respectively)
ARP	Auxiliary register pointer
BR	Branch address
D	Data memory address field
DATn	Label assigned to data memory location n
dma	Data memory address
DP	Data page pointer
INTM	Interrupt mode bit
italics	User-defined terms
K	Immediate operand field
M	Addressing mode bit
*nn*h	Indicates nn is a hexadecimal number. (All others are assumed to be decimal values.)
OVM	Overflow (saturation) mode flag bit
P	Product register
PA	Port address (PA0 through PA7 are predefined assembler symbols equal to 0 through 7, respectively.)
PC	Program counter
pma	Program memory address
PRGn	Label assigned to program memory location n
R	1-bit operand field specifying auxiliary register
S	4-bit left-shift code

Table 4–1. Instruction Symbols (Continued)

Symbol	Meaning
T	Temporary register
TOS	Top of stack
X	3-bit accumulator left-shift field
→	Is assigned to
| |	An absolute value
[ ]	Options
( )	Contents of
{ }	Alternatives, one of which must be entered
< > or ≠	Not equal

4.2.2 Instruction Set Summary

Table 4–2 summarizes the TMS320C1x instruction set according to function and alphabetically within each functional grouping.

The instruction set summary consists primarily of single-cycle, single-word instructions. Only infrequently used branch and I/O instructions are multicycle.

Table 4–2. Instruction Set Summary

Mnemonic and Description		Cycles	Words	16-Bit Opcode MSB			LSB
Accumulator Memory Reference Instructions							
ABS	Absolute value of accumulator	1	1	0111	1111	1000	1000
ADD	Add to accumulator with shift	1	1	0000	SSSS	MDDD	DDDD
ADDH	Add to high accumulator	1	1	0110	0000	MDDD	DDDD
ADDS	Add to low accumulator with sign extension suppressed	1	1	0110	0001	MDDD	DDDD
AND	AND with accumulator	1	1	0111	1001	MDDD	DDDD
LAC	Load accumulator with shift	1	1	0010	SSSS	MDDD	DDDD

Table 4–2. Instruction Set Summary (Continued)

Mnemonic and Description		Cycles	Words	16-Bit Opcode MSB			LSB
Accumulator Memory Reference Instructions							
LACK	Load accumulator immediate short	1	1	0111	1110	KKKK	KKKK
OR	OR with accumulator	1	1	0111	1010	MDDD	DDDD
SACH	Store high accumulator with shift	1	1	0101	1XXX	MDDD	DDDD
SACL	Store low accumulator	1	1	0101	0000	MDDD	DDDD
SUB	Subtract from accumulator with shift	1	1	0001	SSSS	MDDD	DDDD
SUBC	Conditional subtract	1	1	0110	0100	MDDD	DDDD
SUBH	Subtract from high accumulator	1	1	0110	0010	MDDD	DDDD
SUBS	Subtract from low accumulator with sign extension suppressed	1	1	0110	0011	MDDD	DDDD
XOR	Exclusive-OR with low accumulator	1	1	0111	1000	MDDD	DDDD
ZAC	Zero accumulator	1	1	0111	1111	1000	1001
ZALH	Zero low accumulator and load high accumulator	1	1	0110	0101	MDDD	DDDD
ZALS	Zero accumulator and load low accumulator with sign extension suppressed	1	1	0110	0110	MDDD	DDDD
Auxiliary Register and Data Page Pointer Instructions							
LAR	Load auxiliary register	1	1	0011	100R	MDDD	DDDD
LARK	Load auxiliary register immediate short	1	1	0111	000R	KKKK	KKKK
LARP	Load auxiliary register pointer immediate	1	1	0110	1000	1000	000K
LDP	Load data memory page pointer	1	1	0110	1111	MDDD	DDDD
LDPK	Load data memory page pointer immediate	1	1	0110	1110	0000	000K
MAR	Modify auxiliary register	1	1	0110	1000	MDDD	DDDD
SAR	Store auxiliary register	1	1	0011	000R	MDDD	DDDD
T Register, P Register, and Multiply Instructions							
APAC	Add P register to accumulator	1	1	0111	1111	1000	1111
LT	Load T register	1	1	0110	1010	MDDD	DDDD
LTA	Load T register and accumulate previous product	1	1	0110	1100	MDDD	DDDD
LTD	Load T register, accumulate previous product, and move data	1	1	0110	1011	MDDD	DDDD
MPY	Multiply (with T register, store product in P register)	1	1	0110	1101	MDDD	DDDD
MPYK	Multiply immediate	1	1	100K	KKKK	KKKK	KKKK
PAC	Load accumulator with P register	1	1	0111	1111	1000	1110
SPAC	Subtract P register from accumulator	1	1	0111	1111	1001	0000

Table 4–2. Instruction Set Summary (Continued)

Mnemonic and Description		Cycles	Words	16-Bit Opcode MSB			LSB
Branch/Call Instructions							
B	Branch unconditionally	2	2	1111 0000	1001 BBBB	0000 BBBB	0000 BBBB
BANZ	Branch on auxiliary register not zero	2	2	1111 0000	0100 BBBB	0000 BBBB	0000 BBBB
BGEZ	Branch if accumulator ≥ 0	2	2	1111 0000	1101 BBBB	0000 BBBB	0000 BBBB
BGZ	Branch if accumulator > 0	2	2	1111 0000	1100 BBBB	0000 BBBB	0000 BBBB
BIOZ	Branch on I/O status = 0	2	2	1111 0000	0110 BBBB	0000 BBBB	0000 BBBB
BLEZ	Branch if accumulator ≤ 0	2	2	1111 0000	1011 BBBB	0000 BBBB	0000 BBBB
BLZ	Branch if accumulator < 0	2	2	1111 0000	1010 BBBB	0000 BBBB	0000 BBBB
BNZ	Branch if accumulator ≠ 0	2	2	1111 0000	1110 BBBB	0000 BBBB	0000 BBBB
BV	Branch on overflow	2	2	1111 0000	0101 BBBB	0000 BBBB	0000 BBBB
BZ	Branch if accumulator = 0	2	2	1111 0000	1111 BBBB	0000 BBBB	0000 BBBB
CALA	Call subroutine indirect	2	1	0111	1111	1000	1100
CALL	Call subroutine	2	2	1111 0000	1000 BBBB	0000 BBBB	0000 BBBB
RET	Return from subroutine	2	1	0111	1111	1000	1101
Control Instructions							
DINT	Disable interrupt	1	1	0111	1111	1000	0001
EINT	Enable interrupt	1	1	0111	1111	1000	0010
LST	Load status register from data memory	1	1	0111	1011	MDDD	DDDD
NOP	No operation	1	1	0111	1111	1000	0000
POP	Pop top of stack to low accumulator	2	1	0111	1111	1001	1101
PUSH	Push low accumulator onto stack	2	1	0111	1111	1001	1100
ROVM	Reset overflow mode	1	1	0111	1111	1000	1010
SOVM	Set overflow mode	1	1	0111	1111	1000	1011
SST	Store status register	1	1	0111	1100	MDDD	DDDD

Table 4–2. Instruction Set Summary (Continued)

Mnemonic and Description		Cycles	Words	16-Bit Opcode MSB			LSB
I/O and Data Memory Operations							
DMOV	Data move in data memory	1	1	0110	1001	MDDD	DDDD
IN	Input data from port	2	1	0100	0AAA	MDDD	DDDD
OUT	Output data to port	2	1	0100	1AAA	MDDD	DDDD
TBLR	Table read	3	1	0110	0111	MDDD	DDDD
TBLW	Table write	3	1	0111	1101	MDDD	DDDD

4.3 Individual Instruction Descriptions

The remainder of this chapter is a reference. The instructions are organized alphabetically, one instruction per page, with information that includes assembler syntax, operands, execution, encoding, description, words, cycles, and examples for each instruction. An example instruction is provided first to familiarize you with the special format used and to explain its content. Refer to Section 4.1 for further information on memory addressing. Code examples using many of the instructions are given in Chapter 5 on *Software Applications*.The following is an alphabetical table of contents for the instructions reference:

Syntax

Direct: [*label*] EXAMPLE *dma* [*shift*]
Indirect: [*label*] EXAMPLE{*| * + | *–} [,*shift* [,*next ARP*]]
Immediate: [*label*] EXAMPLE [*constant*]

Each instruction begins with an assembler syntax expression. The optional comment field that concludes the syntax is not included in the syntax expression. Space(s) are required between each field (label, command, operand, and comment fields), as shown in the syntax. The syntax example illustrates both direct and indirect addressing, as well as immediate addressing in which the operand field includes *constant*.

Operands

$0 \leq$ dma ≤ 127
ARP = 0 or 1
$0 \leq$ constant ≤ 255

Operands may be constants or assembly-time expressions that refer to memory, I/O and register addresses, pointers, shift counts, and a variety of constants. The operand values used in the example syntax are shown.

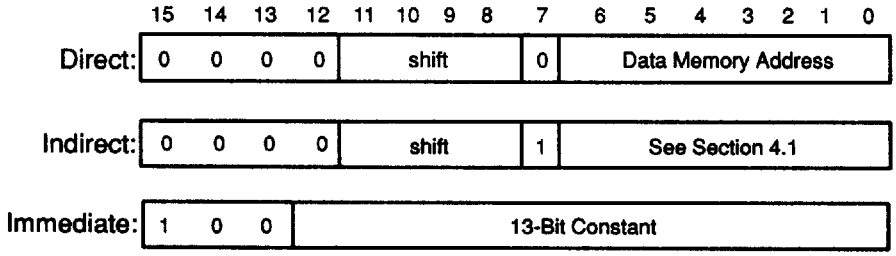
Operation

(PC) + 1 → PC
(ACC) + (dma) 2^{shift} → ACC

1 → interrupt mode (INTM) status bit. Affects INTM.

The operation sequence describes the processing that occurs when the instruction is executed. Conditional effects of status register specified modes are also given. In addition, those bits in the status registers that are affected by the instruction are named.

Encoding



Encoding shows opcode examples of both direct and indirect addressing or immediate addressing.

Description

Description explains the instruction execution and its effect on the rest of the processor or memory contents. Any constraints on the operands imposed by the processor or the assembler are also described here. The description parallels and supplements the information given by the operation section.

Words

1

The digit specifies the number of memory words required to store the instruction and its extension words

Cycles

1

The digit specifies the number of cycles required to execute the instruction.

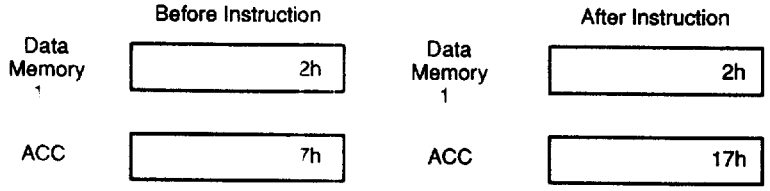
Example

```
ADD  DAT1,3    ;(DP = C)
```

or

Example

```
ADD  *,3       ;If current auxiliary register contains 1
```



The sample code presented in the above format shows the effect of the code on memory and/or registers.

Syntax [*label*] ABS

Operands None

Operation (PC) + 1 → PC
If (ACC) < 0:
Then – (ACC) → ACC

Encoding

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	1	1	1	1	1	1	1	0	0	0	1	0	0	0

Description If the contents of the accumulator are greater than or equal to zero, the accumulator is unchanged by the execution of ABS. If the contents of the accumulator are less than zero, the accumulator is replaced by its 2s-complement value.

Note that 80000000h is a special case. When the overflow mode is not set, the ABS of 80000000h is 80000000h. When the overflow mode is set, the ABS of 80000000h is 7FFFFFFFh.

Words 1

Cycles 1

Example

```
ABS
```

	Before Instruction		After Instruction
ACC	1234h	ACC	1234h
ACC	0FFFFFFFFh	ACC	1h

Syntax

Direct: [*label*] ADD *dma*, [*shift*]
Indirect: [*label*] ADD {*|* * + | *–} [,*shift* [,*next ARP*]]

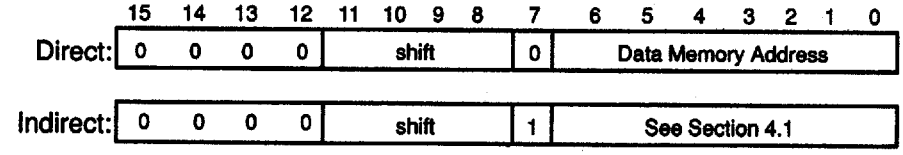
Operands

0 ≤ dma ≤ 127
ARP = 0 or 1
0 ≤ shift ≤15 (defaults to 0)

Operation

(PC) + 1 → PC
(ACC) + (dma) × 2^{shift} → ACC

Encoding



Description

Contents of the addressed data memory location are left-shifted and added to the accumulator. During shifting, low-order bits are zero-filled, and high-order bits are sign extended. The result is stored in the accumulator.

Words

1

Cycles

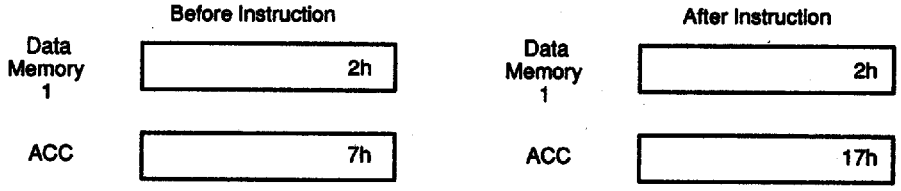
1

Example 1

```
ADD  DAT1,3    ;(DP = 0)
```

or

```
ADD  *,3       ;If current auxiliary register contains 1
```

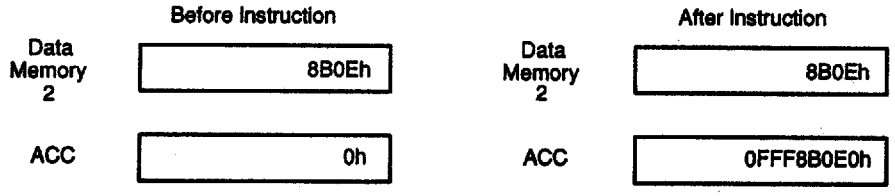


Example 2

```
ADD  DAT2,4    ;(DP = 0)
```

or

```
ADD  *,4       ;If current auxiliary register contains 2
```



Syntax

Direct: [*label*] ADDH *dma*
Indirect: [*label*] ADDH {*|* + | *–} [,*next ARP*]

Operands

$0 \leq dma \leq 127$
ARP = 0 or 1

Operation

(PC) + 1 → PC
(ACC) + (dma) × 2^{16} ACC

Encoding

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Direct:	0	1	1	0	0	0	0	0	0	Data Memory Address						
Indirect:	0	1	1	0	0	0	0	0	1	See Section 4.1						

Contents of the addressed data memory location are added to the upper half of the accumulator (bits 31 through 16). Low-order bits are unaffected by ADDH.

The ADDH instruction may be used in performing 32-bit arithmetic.

Words

1

Cycles

1

Example

```
ADDH    DAT5    ;(DP = 0)
```

or

```
ADDH    *       ;If current auxiliary register contains 5
```

	Before Instruction		After Instruction	
Data Memory 5	4h		Data Memory 5	4h
ACC	13h		ACC	40013h

Syntax

Direct: [*label*] ADDS *dma*
Indirect: [*label*] ADDS {*|*+|*–} [,*next ARP*]

Operands

0 ≤ dma ≤ 127
ARP = 0 or 1

Operation

(PC) + 1 → PC
(ACC) + (dma) → ACC
(dma) is a 16-bit unsigned number.
Affects OV; affected by OVM.

Encoding

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Direct:	0	1	1	0	0	0	0	1	0	Data Memory Address						
Indirect:	0	1	1	0	0	0	0	1	1	See Section 4.1						

Description

Contents of the specified data memory location are added with sign extension suppressed. The data is treated as a 16-bit unsigned number rather than a 2s-complement number. Therefore, there is no sign extension such as there is with the ADD instruction.

The ADDS instruction can be used in implementing 32-bit arithmetic.

Words

1

Cycles

1

Example

```
ADDS DAT11     ;(DP = 0)
```

or

```
ADDS     *     ;If current auxiliary register contains 11
```

	Before Instruction	After Instruction
Data Memory 11	0F006h	0F006h
ACC	3h	0F009h

Syntax

Direct: [*label*] AND *dma*
Indirect: [*label*] AND {*|* * + | *–} [,*next ARP*]

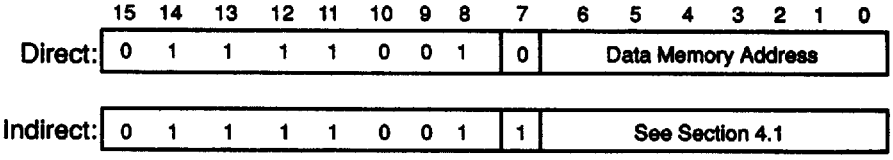
Operands

0 ≤ dma ≤ 127
ARP = 0 or 1

Operation

(PC) + 1 → PC
(ACC(15–0)).AND.(dma) → ACC(15–0)
0 → ACC(31–16)

Encoding



Description

The lower half of the accumulator is ANDed with the contents of the addressed data memory location. The upper half of the accumulator is ANDed with all zeros. Therefore, the upper half of the accumulator is always zeroed by the AND instruction.

Words

1

Cycles

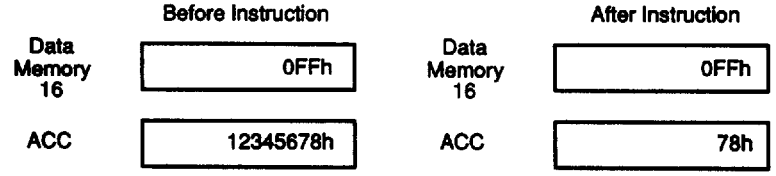
1

Example

```
AND     DAT16    ;(DP = 0)
```

or

```
AND     *        ;If current auxiliary register contains 16
```



Syntax [*label*] APAC

Operands None

Operation (PC) + 1 → PC
(ACC) + (P register) → ACC
Affects OV; affected by OVM.

Encoding

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	1	1	1	1	1	1	1	0	0	0	1	1	1	1

Description The contents of the P register, the result of a multiply, are added to the contents of the accumulator. The result is stored in the accumulator. The APAC instruction is a subset of the LTA and LTD instructions.

Words 1

Cycles 1

Example

```
APAC
```

	Before Instruction		After Instruction
P	40h	P	40h
ACC	20h	ACC	60h

Syntax [*label*] B *pma*

Operands 0 ≤ pma ≤ 4095

Operation pma → PC

Encoding

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	1	1	0	0	1	0	0	0	0	0	0	0	0
Program Memory Address															

Description Control passes to the designated program memory address (pma). Pma can be either a symbolic or a numeric address.

Words 2

Cycles 2

Example

```
B    PRG191    ;191 is loaded into the program counter and
               ;the program continues running from that
               ;location
```

Syntax [*label*] BANZ *pma*

Operands 0 ≤ pma ≤ 4095

Operation

If (AR bits 8–0) ≠ 0:
Then pma → PC;
Else (PC) + 2 → PC
(AR) – 1 → AR.

Encoding

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	1	0	1	0	0	0	0	0	0	0	0	0	0
Program Memory Address															

Description If the lower nine bits of the current auxiliary register are not equal to zero, then the address contained in the following word is loaded into the program counter. If these bits are equal to zero, the current program counter is incremented by two. In either case, the auxiliary register is decremented. Note that the test for zero is performed before decrementing the auxiliary register. The branch to a location in program is specified by the program memory address (pma). Pma can be either a symbolic or numeric address.

Words 2

Cycles 2

Example

```
BANZ    PRG35
```

	Before Instruction		After Instruction
AR	1h	AR	0h
PC	46h	PC	35h

or

AR	0h	AR	0FFFFh
PC	46h	PC	48h

Note:

BANZ is designed for loop control using the auxiliary registers as loop counters. The auxiliary register is decremented after testing for zero. The auxiliary registers also behave as modulo 512 counters.

Syntax [*label*] BGEZ *pma*

Operands 0 ≤ pma ≤ 4095

Operation If (ACC) ≥ 0:
Then pma → PC;
Else (PC) + 2 → PC

Encoding

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	1	1	1	0	1	0	0	0	0	0	0	0	0
Program Memory Address															

If the contents of the accumulator are greater than or equal to zero, then branch to the specified program memory location. The branch to a location in program is specified by the program memory address (pma). Pma can be either a symbolic or numeric address.

Words 2

Cycles 2

Example

```
BGEZ     PRG217   ;217 is loaded into the program counter
                  ;if the accumulator is greater than or
                  ;equal to zero
```

Syntax [*label*] BGZ *pma*

Operands 0 ≤ pma ≤ 4095

Operation If (ACC) > 0:
Then pma → PC;
Else (PC) + 2 → PC.

Encoding

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0
Program Memory Address															

If the contents of the accumulator are greater than zero, then branch to the specified program memory location. The branch to a location in program is specified by the program memory address (pma). Pma can be either a symbolic or numeric address.

Words 2

Cycles 2

Example

```
BGZ     PRG342   ;342 is loaded into the program counter
                 ;if the accumulator is greater than zero
```

Syntax

[*label*] BIOZ *pma*

Operands

$0 \leq pma \leq 4095$

Operation

If $\overline{BIO} = 0$:
Then pma → PC;
Else (PC) + 2 → PC.

Encoding

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	1	0	1	1	0	0	0	0	0	0	0	0	0
Program Memory Address															

Description

If the $\overline{BIO}$ pin is active low, then branch to the specified program memory location. Otherwise, the program counter is incremented to the next instruction. The branch to a location in program is specified by the program memory address (pma). Pma can be either a symbolic or numeric address.

The BIOZ instruction in conjunction with the $\overline{BIO}$ pin can be used to test whether a peripheral is ready to send or receive data. Polling the $\overline{BIO}$ pin by using BIOZ may be preferable to an interrupt when executing time-critical loops.

Words

2

Cycles

2

Example

```
BIOZ     PRG64    ;If the BIO pin is active (low), then
                  ;a branch to location 64 occurs.
                  ;Otherwise, the program counter is
                  ;incremented
```

Note:

The BIOZ instruction is not defined on the TMS320C14 devices.

Syntax [*label*] BLEZ *pma*

Operands 0 ≤ pma ≤ 4095

Operation If (ACC) ≤ 0:
Then pma → PC;
Else (PC) + 2 → PC.

Encoding

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	1	1	0	1	1	0	0	0	0	0	0	0	0
Program Memory Address															

Description If the contents of the accumulator are less than or equal to zero, then branch to the specified program memory location. The branch to a location in program is specified by the program memory address (pma). Pma can be either a symbolic or a numeric address

Words 2

Cycles 2

Example

```
BLEZ     PRG63    ;63 is loaded into the program counter if
                  ;the accumulator is less than or equal to
                  ;zero
```

Syntax [*label*] BLZ *pma*

Operands 0 ≤ pma ≤ 4095

Operation If (ACC) < 0:
Then pma → PC;
Else (PC) + 2 → PC.

Encoding

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	1	1	0	1	0	0	0	0	0	0	0	0	0
Program Memory Address															

Description If the contents of the accumulator are less than zero, then branch to the specified program memory location. The branch to a location in program is specified by the program memory address (pma). Pma can be either a symbolic or a numeric address.

Words 2

Cycles 2

Example

```
BLZ      PRG481   ;481 is loaded into the program counter
                  ;if the accumulator is less than zero
```

Syntax [*label*] BNZ *pma*

Operands 0 ≤ pma ≤ 4095

Operation If (ACC) ≠ 0:
Then pma → PC;
Else (PC) + 2 → PC.

Encoding

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0
Program Memory Address															

Description If the contents of the accumulator are not equal to zero, then branch to the specified program memory location. The branch to a location in program is specified by the program memory address (pma). Pma can be either a symbolic or a numeric address.

Words 2

Cycles 2

Example

```
BNZ       PRG320    ;320 is loaded into the program counter
                    ;if the accumulator does not equal zero
```

Syntax [*label*] BV *pma*

Operands 0 ≤ pma ≤ 4095

Operation If overflow (OV) status bit = 1:
Then pma → PC and 0 → OV;
Else (PC) + 2 → PC.
Affects OV; affected by OV.

Encoding

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	1	0	1	0	1	0	0	0	0	0	0	0	0
Program Memory Address															

Description If the overflow (OV) flag has been set, then a branch to the specified program memory location occurs, and the overflow flag is cleared. Otherwise, the program counter is incremented to the next instruction. The branch to a location in program is specified by the program memory address (pma). Pma can be either a symbolic or numeric address.

Words 2

Cycles 2

Example

```
BV      PRG610   ;If an overflow has occurred since the
                 ;overflow flag was last cleared, then 610
                 ;is loaded into the program counter and
                 ;OV is cleared. Otherwise, the program
                 ;counter is incremented
```

Syntax [*label*] BZ *pma*

Operands 0 ≤ pma ≤ 4095

Operation If (ACC) = 0:
Then pma → PC;
Else (PC) + 2 → PC.

Encoding

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0
Program Memory Address															

Description If the contents of the accumulator are equal to zero, then branch to the specified program memory location. The branch to a location in program is specified by the program memory address (pma). Pma can be either a symbolic or numeric address.

Words 2

Cycles 2

Example

```
BZ   PRG102   ;102 is loaded into the program counter if the
              ;accumulator is equal to zero
```

Syntax [*label*] CALA

Operands None

Operation (PC) + 1 → TOS
(ACC(11–0)) → PC

Encoding

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	1	1	1	1	1	1	1	0	0	0	1	1	0	0
Program Memory Address															

Description The current program counter is incremented and pushed onto the top of the stack. Then, the contents of the 12 least significant bits of the accumulator are loaded into the PC.

The CALA instruction is used to perform computed subroutine calls.

Words 1

Cycles 2

Example

```
CALA
```

	Before Instruction		After Instruction
PC	25h	PC	83h
ACC	83h	ACC	83h
Stack	32h 75h 84h 49h	Stack	26h 32h 75h 84h

Syntax [*label*] CALL *pma*

Operands 0 ≤ pma ≤ 4095

Operation (PC) + 2 → TOS
pma → PC

Encoding

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0
Program Memory Address															

Description The current program counter is incremented by two and pushed onto the top of the stack. The specified program memory address (pma) is then loaded into the PC. Pma can be either a symbolic or a numeric address.

Words 2

Cycles 2

Example

```
CALL    PRG109
```

	Before Instruction		After Instruction
PC	33h	PC	6Dh
Stack	71h 48h 16h 80h	Stack	35h 71h 48h 16h

Syntax [*label*] DINT

Operands None

Operation (PC) + 1 → PC
1 → interrupt mode (INTM) status bit.
Affects INTM.

Encoding

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	1	1	1	1	1	1	1	0	0	0	0	0	0	1

Description The interrupt mode (INTM) status bit is set to logic 1. Maskable interrupts are disabled immediately after the DINT instruction executes. Interrupts are also disabled by a reset. Note that the LST instruction does ***not*** affect INTM.

The unmaskable interrupt, $\overline{RS}$, is not disabled by this instruction.

Words 1

Cycles 1

Example

```
DINT    ;Maskable interrupts are disabled, and INTM
        ;is set to one
```

Syntax

Direct: [*label*] DMOV *dma*
Indirect: [*label*] DMOV {*|* * + | *–} [,*next ARP*]

Operands

0 ≤ dma ≤ 127
ARP = 0 or 1

Operation

(PC) + 1 → PC
(dma) → dma + 1

Encoding

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Direct:	0	1	1	0	1	0	0	1	0	Data Memory Address						
Indirect:	0	1	1	0	1	0	0	1	1	See Section 4.1						

Description

The contents of the specified data memory address are copied into the contents of the next higher address. When data is copied from the addressed location to the next higher location, the contents of the addressed location remain unaltered.

The data move function is useful in implementing the z^{-1} delay encountered in digital signal processing. The DMOV function is included in the LTD instruction (see LTD for more information).

Words

1

Cycles

1

Example

```
DMOV    DAT8
```

or

```
DMOV    *       ;If current auxiliary register contains 8
```

	Before Instruction	After Instruction
Data Memory 8	43h	43h
Data Memory 9	2h	43h

Syntax [*label*] EINT

Operands None

Operation (PC) + 1 → PC
0 → interrupt mode (INTM) status bit
Affects INTM.

Encoding

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	1	1	1	1	1	1	1	0	0	0	0	0	1	0

Description The interrupt mode (INTM) status bit is cleared to logic 0. Maskable interrupts are enabled after the instruction following EINT executes. This allows an interrupt service routine to re-enable interrupts and execute a RET instruction before any other pending interrupts are processed. Note that the EINT instruction should not be used immediately preceding a branch instruction.

The LST instruction does not affect INTM. (See the DINT instruction for further information.)

Words 1

Cycles 1

Example

```
EINT        ;Maskable interrupts are enabled, and INTM
            ;is set to zero
```

Syntax

Direct: [*label*] IN *dma PA*
Indirect: [*label*] IN {*|* + | *–}, *PA* [,*next ARP*]

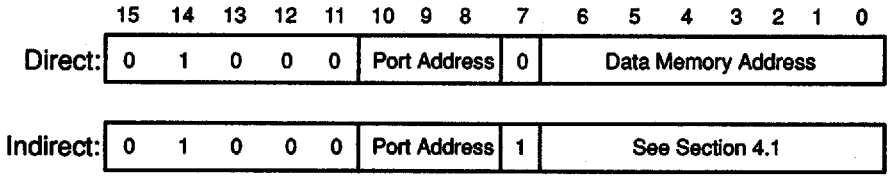
Operands

0 ≤ dma ≤ 127
ARP = 0 or 1
0 ≤ port address PA ≤ 7

Operation

(PC) + 1 → PC
Port address → address lines A2/PA2–A0/PA0
0 → address bus A11–A3
Data bus D15–D0 → dma

Encoding



Description

The IN instruction reads data from a peripheral and places it in data memory. This is a two-cycle instruction. During the first cycle, the port address is sent to address lines A2/PA2–A0/PA0. $\overline{DEN}$ goes low during the same cycle, strobing in the data that the addressed peripheral places on the data bus D15–D0. On the TMS320C10, and on the 'C15, $\overline{MEN}$ remains high when $\overline{DEN}$ is active. On the TMS320C17, the $\overline{MEN}$ signal is not available.

Note:

On the TMS320C14, address lines A11–A4 are driven high during an I/O access. These address lines are driven low for all other devices.

Words

1

Cycles

2

Example

```
IN    STAT,PA5     ;Read in word from peripheral on port
                   ;address 5. Store in data memory
                   ;location STAT
```

or

```
LARK  1,20         ;Load AR1 with decimal 20
LARP  1            ;Load ARP with decimal 1
IN    *-,PA1,0     ;Read in word from peripheral on port
                   ;address 1. Store in data memory
                   ;location 20. Decrement AR1 to 19
                   ;Load the ARP with 0
```

Syntax

Direct: [*label*] LAC *dma* [,*shift*]
Indirect: [*label*] LAC {*|* * + | *–} [,*shift* [,*next ARP*]]

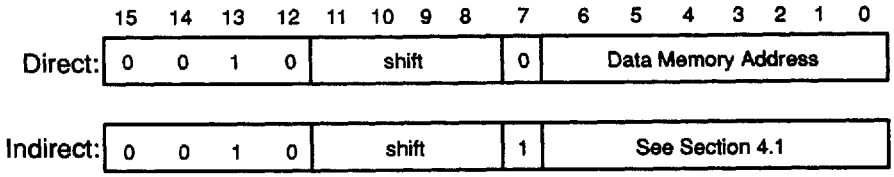
Operands

0 ≤ dma ≤ 127
ARP = 0 or 1
0 ≤ shift ≤ 15 (defaults to 0)

Operation

(PC) + 1 → PC
(dma) × 2^{shift} → ACC

Encoding



Description

Contents of the specified data memory address are left-shifted and loaded into the accumulator. During shifting, low-order bits are zero-filled. High-order bits are sign-extended.

Words

1

Cycles

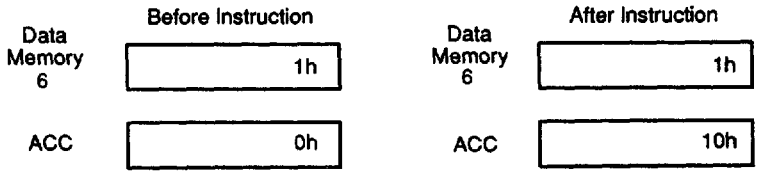
1

Example

```
LAC  DAT6,4          ;(DP = 0)
```

or

```
LAC  *,4             ;If current auxiliary register contains 6
```



Syntax [*label*]LACK *constant*

Operands 0 ≤ constant ≤ 255

Operation (PC) + 1 → PC
8-bit positive constant → ACC

Encoding

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	1	1	1	1	1	0	8-Bit Constant							

Description The 8-bit constant is loaded into the accumulator right-justified. The upper 24 bits of the accumulator are zeroed (that is, sign extension is suppressed).

Words 1

Cycles 1

Example

```
LACK    15h
```

Syntax

Direct: [*label*] LAR *AR, dma*
Indirect: [*label*] LAR *AR*, {*|* * + | *–} [,*next ARP*]

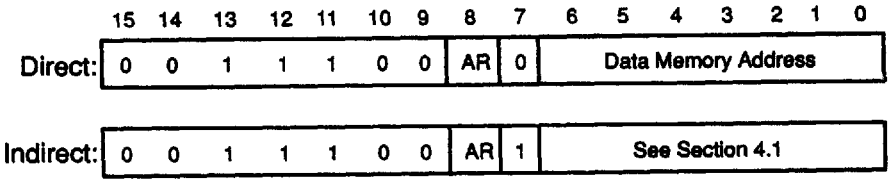
Operands

0 ≤ dma ≤ 127
auxiliary register AR = 0 or 1
ARP = 0 or 1

Operation

(PC) + 1 → PC
(dma) → auxiliary register AR

Encoding



Description

The contents of the specified data memory address are loaded into the designated auxiliary register. The LAR and SAR (store auxiliary register) instructions can be used to load and store the auxiliary registers during subroutine calls and interrupts. If an auxiliary register is not being used for indirect addressing, LAR and SAR enable the register to be used as an additional storage register, especially for swapping values between data memory locations without affecting the contents of the accumulator.

If indirect addressing is used to load the current auxiliary register (that is, if the AR specified in the LAR instruction is the AR pointed to by the ARP), then the new value is loaded into the auxiliary register from data memory, and any decrement or increment specified is not performed.

Words

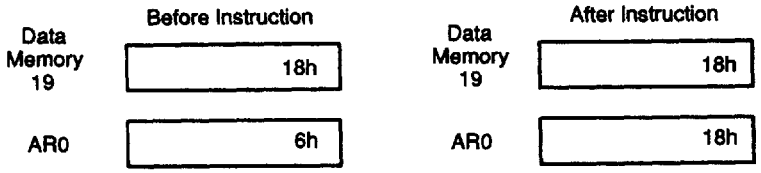
1

Cycles

1

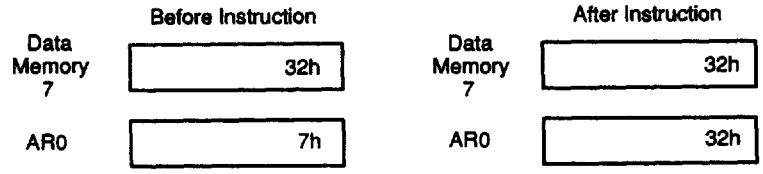
Example

```
LAR     AR0,DAT19
```



also,

```
LARP    0
LAR     AR0,*-
```



Syntax [*label*] LARK *AR, constant*

Operands 0 ≤ constant ≤ 255
auxiliary register AR = 0 or 1

Operation (PC) + 1 → PC
8-bit constant → auxiliary register AR

Encoding

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	1	1	0	0	0	AR	8-Bit Constant							

Description The 8-bit positive constant is loaded into the designated auxiliary register, right-justified and zero-filled (that is, sign extension suppressed).

LARK is useful for loading an initial loop counter value into an auxiliary register for use with the BANZ instruction.

Words 1

Cycles 1

Example

```
LARK    AR0,21h
```

	Before Instruction		After Instruction
AR0	0h	AR0	21h

Syntax [*label*] LARP *constant*

Operands 0 ≤ constant ≤ 1

Operation (PC) + 1 → PC
Constant → ARP
Affects ARP.

Encoding

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	1	0	1	0	0	0	1	0	0	0	0	0	0	ARP

The auxiliary register pointer is loaded with the one-bit constant identifying the desired auxiliary register. ARP can also be modified by the LST and MAR instructions, as well as any instruction that is used in the indirect addressing mode.

The LARP instruction is a subset of MAR; that is, the opcode is the same as MAR in the indirect addressing mode. The instruction MAR *,*next ARP* has the same effect as LARP.

Words 1

Cycles 1

Example

```
LARP 1          ;Any succeeding instructions will use auxiliary
                ;register AR1 for indirect addressing
```

Syntax

Direct: [*label*] LDP *dma*
Indirect: [*label*] LDP {*|* + |*–} [,*next ARP*]

Operands

0 ≤ dma ≤ 127
ARP = 0 or 1

Operation

(PC) + 1 → PC
LSB of (dma) → data memory page pointer (DP = 0 or 1)
Affects DP.

Encoding

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Direct:	0	1	1	0	1	1	1	1	0	Data Memory Address						
Indirect:	0	1	1	0	1	1	1	1	1	See Section 4.1						

Description

The least significant bit of the contents of the specified data memory address is loaded into the DP (data memory page pointer) register. All higher-order bits are ignored in the data word. DP = 0 defines page 0 that contains words 0–127. DP=1 defines page 1 that contains words 128–143/255.The DP may also be loaded by the LST and LDPK instructions.

Words

1

Cycles

1

Example

```
LDP  DAT1      ;LSB of location DAT1 is loaded into DP
```

or

```
LDP  *,1       ;LSB of location currently addressed by auxiliary
               ;register is loaded into DP. ARP is set to 1
```

	Before Instruction		After Instruction	
Data Memory 1	0FEDCh		Data Memory 1	0FEDCh
DP	1h		DP	0h

Syntax [*label*] LDPK *constant*

Operands 0 ≤ constant ≤ 1

Operation (PC) + 1 → PC
Constant → data memory page pointer (DP)
Affects DP.

Encoding

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	1	0	1	1	1	0	0	0	0	0	0	0	0	DP

Description The DP (data memory page pointer) register is loaded with a 1-bit constant. DP = 0 defines page 0 that contains words 0–127. DP = 1 defines page 1 that contains words 128–143/255. The DP can also be loaded by the LST and LDP instructions.

Words 1

Cycles 1

Example

```
LDPK    0    ;The data pointer is set to 0
```

Syntax

Direct: [*label*] LST *dma*
Indirect: [*label*] LST {*|* * - | * -} [,*next ARP*]

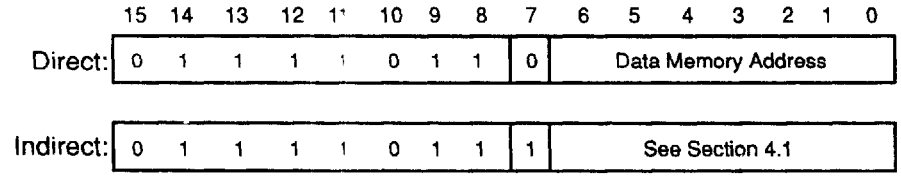
Operands

0 ≤ dma ≤ 127
ARP = 0 or 1

Operation

(PC) + 1 → PC
(dma) → status register bits
Affects ARP, OV, OVM, and DP.
Does not affect INTM.

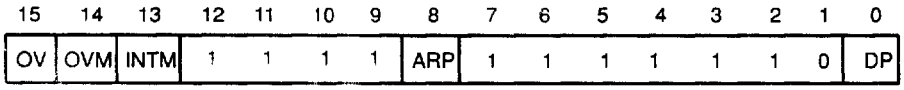
Encoding



Description

The status register is loaded with the addressed data memory value. Note that the INTM (interrupt mode) bit is unaffected by LST.

The LST instruction is used to load the status register after interrupts and subroutine calls. The status register contains the status bits: OV (overflow flag), OVM (overflow mode), ARP (auxiliary register pointer), and DP (data memory page pointer). These bits were stored (by the SST instruction) in the data memory word as follows:



Words

1

Cycles

1

Example

```
LARP    0
```

or

```
LST     *,1  ;The data memory word addressed by the contents
             ;of auxiliary register AR0 replaces the status
             ;bits. ARP becomes 1
```

Note:

When you use direct addressing, the SST instruction always saves status on page 1. The LST instruction will not automatically restore status from page 1. Therefore, you must specify the correct data page pointer.

Syntax

Direct: [*label*] LT *dma*
Indirect: [*label*] LT {*| * + | *–} [,*next ARP*]

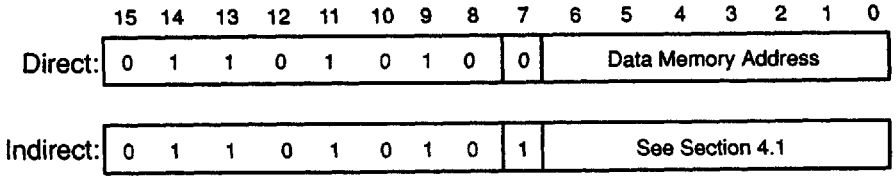
Operands

0 ≤ dma ≤ 127
ARP = 0 or 1

Operation

(PC) + 1 → PC
(dma) → T register

Encoding



Description

The T register is loaded with the contents of the specified data memory location. The LT instruction may be used to load the T register in preparation for multiplication (see the LTA, LTD, MPY, and MPYK instructions).

Words

1

Cycles

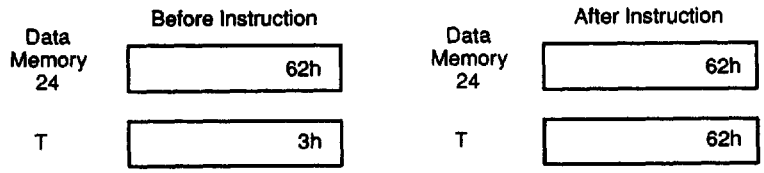
1

Example

```
LT   DAT24     ;(DP = 0)
```

or

```
LT   *         ;If current auxiliary register contains 24
```



Syntax

Direct: [*label*] LTA *dma*
Indirect: [*label*] LTA {*|* + |*–} [,*next ARP*]

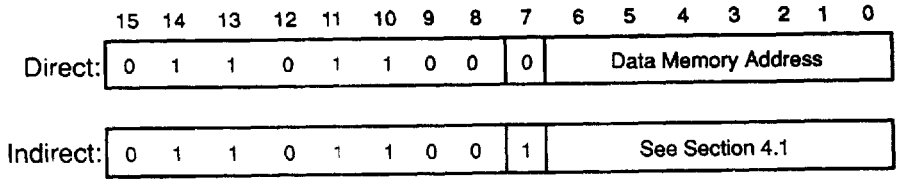
Operands

$0 \leq dma \leq 127$
ARP = 0 or 1

Operation

(PC) + 1 → PC
(dma) → T register
(ACC) + (P register) → ACC
Affects OV; affected by OVM.

Encoding



Description

The T register is loaded with the contents of the specified data memory address. The P register, containing the previous product of the multiply operation, is added to the accumulator, and the result is stored in the accumulator.

The function of the LTA instruction is included in the LTD instruction.

Words

1

Cycles

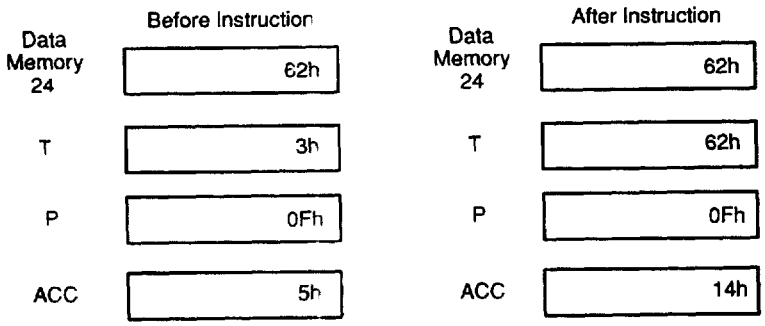
1

Example

```
LTA  DAT24     ;(DP = 0)
```

or

```
LTA  *         ;If current auxiliary register contains 24
```



Syntax

Direct: [*label*] LTD *dma*
Indirect: [*label*] LTD {*|* * + | *–} [,*next ARP*]

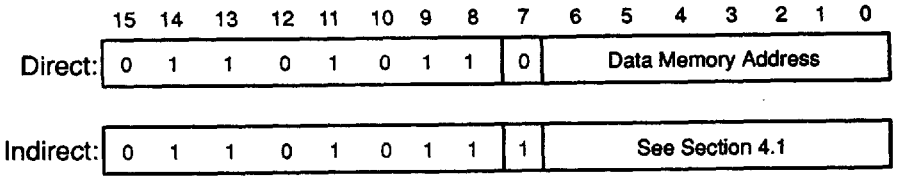
Operands

0 ≤ dma ≤ 127
ARP = 0 or 1

Operation

(PC) + 1 → PC
(dma) → T register
(dma) → dma + 1
(ACC) + (P register) → ACC
Affects OV; affected by OVM.

Encoding



Description

The T register is loaded with the contents of the specified data memory address. The contents of the P register are added to the accumulator, and the result is placed in the accumulator. The contents of the specified data memory address are also copied to the next higher data memory address. This function is described under the instruction DMOV.

Words

1

Cycles

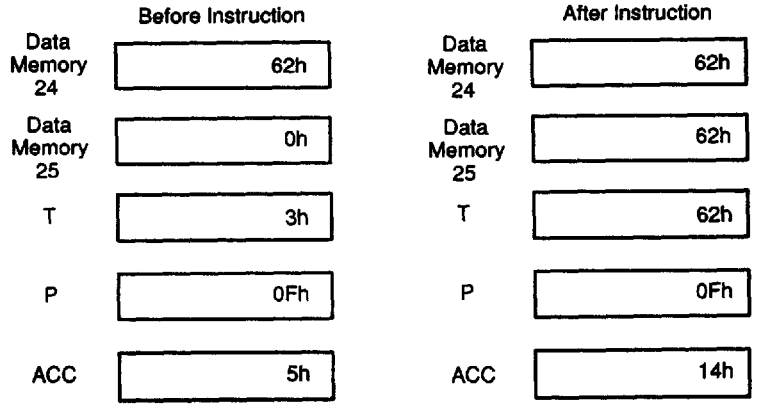
1

Example

```
LTD  DAT24     ;(DP = 0)
```

or

```
LTD  *         ;If current auxiliary register contains 24
```



Syntax

Direct: [*label*] MAR *dma*
Indirect: [*label*] MAR {*|* * + | *–} [,*next ARP*]

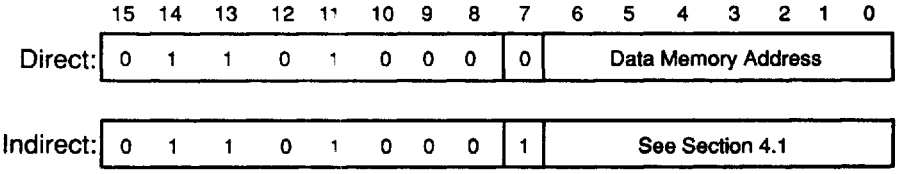
Operands

0 ≤ dma ≤ 127
ARP = 0 or 1

Operation

(PC) + 1 → PC
Modifies AR(ARP), ARP as specified by the indirect addressing field (acts as a NOP in direct addressing).

Encoding



Description

In the indirect addressing mode, the auxiliary registers are either incremented or decremented and the ARP is modified; however, no use is made of the memory being referenced. MAR is used only to modify the auxiliary registers or the ARP. ARP may also be loaded by an LST instruction.

MAR acts as a no-operation (NOP) instruction in the direct addressing mode. Also, the LARP instruction is a subset of MAR (that is, MAR *,0 performs the same function as LARP 0).

Words

1

Cycles

1

Example 1

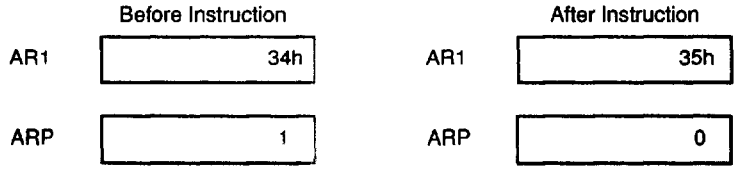
```
MAR  *,1  ;Load the ARP with 1
```

Example 2

```
MAR  *-   ;Decrement current auxiliary register (in this
          ;case, AR1)
```

Example 3

```
MAR  *+,0 ;Increment current auxiliary register (AR1)
          ;and load ARP with 0
```



Syntax

Direct: [*label*] MPY *dma*
Indirect: [*label*] MPY {*|* + | *–} [*next ARP*]

Operands

$0 \leq dma \leq 127$
ARP = 0 or 1

Operation

(PC) + 1 → PC
(T register) × (dma) → P register

Encoding

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Direct:	0	1	1	0	1	1	0	1	0	Data Memory Address						
Indirect:	0	1	1	0	1	1	0	1	1	See Section 4.1						

Description

The contents of the T register are multiplied by the contents of the addressed data memory location. The result is placed in the P register.

During an interrupt, all registers except the P register can be saved and restored directly. However, the TMS320C1x generation TMS320 devices have hardware protection against servicing an interrupt between an MPY or MPYK instruction and the succeding instruction. For this reason, it is advisable to follow MPY and MPYK with LTA, LTD, PAC, APAC, or SPAC.

Note that no provisions are made for the condition of 8000h × 8000h. If this condition arises, the product will be 0C0000000h.

Words

1

Cycles

1

Example

```
MPY     DAT13    ;(DP = 0)
```

or

```
MPY  *           ;If current auxiliary register contains 13
```

	Before Instruction	After Instruction
Data Memory 13	7h	7h
T	6h	6h
P	36h	2Ah

Syntax [*label*] MPYK *constant*

Operands $-2^{12} \leq$ constant $< 2^{12}$

Operation (PC) + 1 → PC
(T register) × constant → P register

Encoding

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	0	0	13-Bit Constant												

Description The contents of the T register are multiplied by the signed 13-bit constant. The result is loaded into the P register.

During an interrupt, all registers except the P register can be saved and restored directly. Because no provision is made to save the contents of the P register during an interrupt, the MPYK instruction should be followed by one of these instructions: PAC, APAC, SPAC, LTA, or LTD. Provision is made in hardware to inhibit the interrupt during MPYK until the next instruction is executed.

Words 1

Cycles 1

Example

```
MPYK    -9
```

	Before Instruction		After Instruction	
T	7h		T	7h
P	2Ah		P	0FFFFFFC1h

Syntax [*label*] NOP

Operands None

Operation (PC) + 1 → PC

Encoding

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0

Description No operation is performed. NOP affects only the PC. NOP is useful as a pad or temporary instruction during program development.

Words 1

Cycles 1

Example

```
NOP
```

Syntax

Direct: [*label*] OR *dma*
Indirect: [*label*] OR {*|* * + |*–} [,*next ARP*]

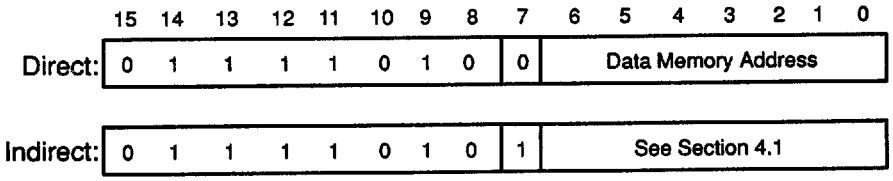
Operands

0 ≤ dma ≤ 127
ARP = 0 or 1

Operation

(PC) + 1 → PC
(ACC(15–0)) .OR.dma → ACC(15–0)
(ACC(31–16)) → ACC(31–16)

Encoding



Description

The low-order bits of the accumulator are ORed with the contents of the addressed data memory location. The high-order bits of the accumulator are ORed with all zeros. Therefore, the upper half of the accumulator is unaffected by this instruction. The result is stored in the accumulator.

The OR instruction is useful for comparing selected bits of a data word.

Words

1

Cycles

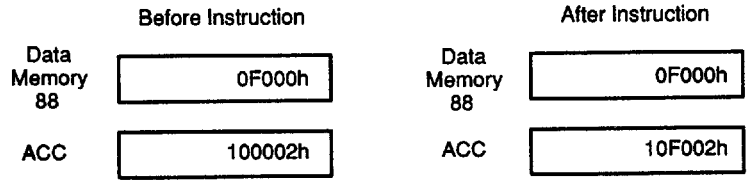
1

Example

```
OR   DAT88     ;(DP = 0)
```

or

```
OR   *         ;Where current auxiliary register contains 88
```



Syntax

Direct: [*label*] OUT*dma, PA*
Indirect: [*label*] OUT {*|* + |*–} *PA* [,*next ARP*]

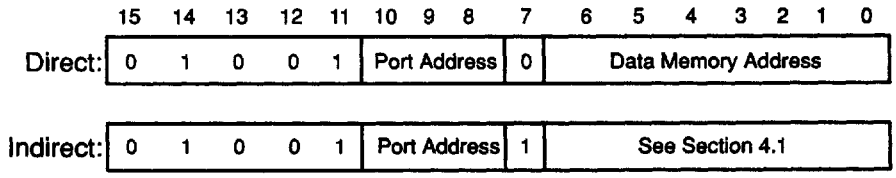
Operands

0 ≤ dma ≤ 127
ARP = 0 or 1
0 ≤ port address PA ≤ 7

Operation

(PC) + 1 → PC
Port address PA → address bus A2/PA2–A0/PA0
0 → address bus A11–A3
(dma) → data bus D15–D0

Encoding



Description

The OUT instruction transfers data from data memory to an external peripheral. The first cycle of this instruction places the port address onto address lines A2/PA2–A0/PA0. During the same cycle, $\overline{WE}$ goes low, and the data word is placed on the data bus D15–D0. On the TMS320C10 and TMS320C15, $\overline{MEN}$ remains high during the first cycle. On the TMS320C17, the $\overline{MEN}$ signal is not available.

Words

1

Cycles

2

Example

```
OUT  120,7     ;Output data word stored in data memory
               ;location 120 to peripheral on port
               ;address 7
```

or

```
OUT  *,5       ;Output data word referenced by current
               ;auxiliary register to peripheral on port
               ;address 5
```

Note:

On the TMS320C14, address lines A11–A4 are driven high during an I/O access. These address lines are driven low for all other devices.

Syntax [*label*] PAC

Operands None

Operation (PC) + 1 → PC
(P register) → ACC

Encoding

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	1	1	1	1	1	1	1	0	0	0	1	1	1	0

Description The contents of the P register resulting from a multiply are loaded into the accumulator.

Words 1

Cycles 1

Example

```
PAC
```

	Before Instruction		After Instruction
P	144h	P	144h
ACC	23h	ACC	144h

Syntax

[*label*] POP

Operands

None

Operation

(PC) + 1 → PC
(TOS) → ACC(11–0)
0 → ACC(31–12)
Pop stack one level.

Encoding

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	1	1	1	1	1	1	1	0	0	1	1	1	0	1

Description

The contents of the top of the stack (TOS) are copied to the low accumulator, and the stack is popped after the contents are copied. The next element on the stack becomes the top of the stack. The upper bits (31–12) of the accumulator are zeroed. The hardware stack is a last-in, first-out stack with four locations. Any time a pop occurs, every stack value is copied to the next higher stack location, and the top value is removed from the stack. After a pop, the bottom two stack words have the same value. Each stack value is copied; therefore, if more than three pops (due to POP or RET instructions) occur before any pushes occur, all levels of the stack contain the same value.

Words

1

Cycles

2

Example

```
POP
```

	Before Instruction		After Instruction
ACC	82h	ACC	45h
Stack	45h 16h 7h 33h	Stack	16h 7h 33h 33h

Note:

On the TMS320C16, a 16-bit 8-level stack is implemented to effectively use the device's larger memory reach of 64K words.

Syntax

[*label*] PUSH

Operands

None

Operation

(PC) + 1 → PC
Push all stack locations down one level.
(ACC(11–0)) → TOS

Encoding

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	1	1	1	1	1	1	1	0	0	1	1	1	0	0

Description

The contents of the lower 12 bits (11–0) of the accumulator are copied onto the top of the hardware stack. The stack is pushed down before the accumulator value is copied. The hardware stack is a last-in, first-out stack with four locations. If more than four pushes (due to CALA, CALL, PUSH, TBLR, or TBLW instructions or interrupts) occur before a pop, the first data values written are lost with each succeeding push.

Words

1

Cycles

2

Example

```
PUSH
```

	Before Instruction		After Instruction
ACC	7h	ACC	7h
Stack	2h 5h 3h 0h	Stack	7h 2h 5h 3h

Note:

On the TMS320C16, a 16-bit 8-level stack is implemented to effectively use the device's larger memory reach of 64K words.

Syntax [*label*] RET

Operands None

Operation (TOS) → PC
Pop stack one level.

Encoding

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	1	1	1	1	1	1	1	0	0	0	1	1	0	1

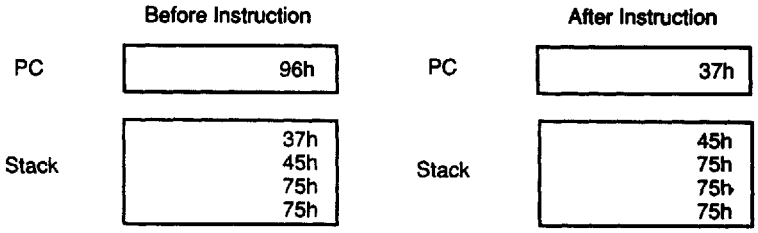
Description The contents of the top of the stack are copied into the program counter. The stack is then popped one level. RET is used in conjunction with CALA and CALL for subroutines and interrupts.

Words 1

Cycles 2

Example

```
RET
```



Syntax [*label*] ROVM

Operands None

Operation (PC) + 1 → PC
0 → OVM status bit
Affects OVM.

Encoding

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	1	1	1	1	1	1	1	0	0	0	1	0	1	0

Description The OVM status bit is reset to logic zero. This disables the overflow mode, in which the device was placed by the SOVM instruction. If an overflow occurs with OVM reset, the OV (overflow flag) is set, and the overflowed result is placed in the accumulator. OVM may also be loaded by the LST and SOVM instructions (see the SOVM instruction).

Words 1

Cycles 1

Example

```
ROVM        ;The overflow mode bit OVM is reset, disabling
            ;the overflow mode on any subsequent arithmetic
            ;operations
```

Syntax

Direct: [*label*] SACH *dma, shift*
Indirect: [*label*] SACH {*|* + | *–} [,*shift* [,*next ARP*]]

Operands

$0 \leq dma \leq 127$
ARP = 0 or 1
shift = 0, 1, or 4

Operation

(PC) + 1 → PC
16 MSBs of (ACC) × 2^{shift} → dma

Encoding

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Direct:	0	1	0	1	1	Shift			0	Data Memory Address						
Indirect:	0	1	0	1	1	Shift			1	See Section 4.1						

Description

The SACH instruction copies the entire accumulator into a shifter. It then left-shifts this entire 32-bit number 0, 1, or 4 bits, and copies the upper 16 bits of the shifted value into data memory. The accumulator itself remains unaffected.

Words

1

Cycles

1

Example

```
SACH    DAT70,1  ;(DP = 0)
```

or

```
SACH    *,1      ;If current auxiliary register contains 70
```

	Before Instruction	After Instruction
ACC	4208001h	4208001h
Data Memory 70	0h	841h

Syntax

Direct: [*label*] SACL *dma*
Indirect: [*label*] SACL {*|* * + | *–} [,*shift* [,*next ARP*]]

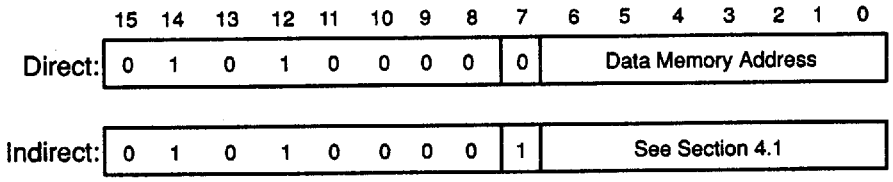
Operands

0 ≤ dma ≤ 127
ARP = 0 or 1
shift = 0

Operation

(PC) + 1 → PC
(ACC(15–0)) → dma

Encoding



Description

The low-order bits of the accumulator are stored in data memory. There is no shift associated with this instruction, although a shift code of zero *must* be specified if the ARP is to be changed.

Words

1

Cycles

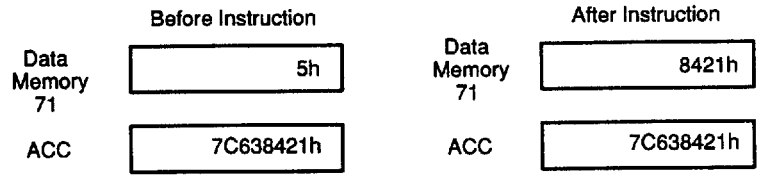
1

Example

```
SACL    DAT71    ;(DP = 0)
```

or

```
SACL    *        ;If current auxiliary register contains 71
```



Syntax

Direct: [*label*] SAR *AR, dma*
Indirect: [*label*] SAR *AR*, {*|* * + | *–} [,*next ARP*]

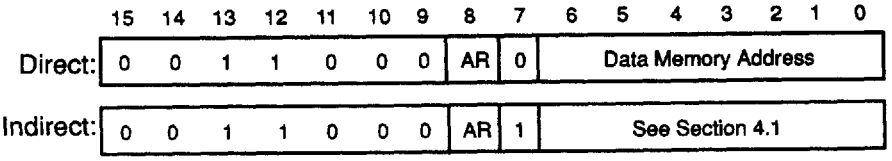
Operands

0 ≤ dma ≤ 127
auxiliary register AR = 0 or 1
ARP = 0 or 1

Operation

(PC) + 1 → PC
(auxiliary register AR) → dma

Encoding



Description

The contents of the designated auxiliary register are stored in the addressed data memory location. For more information, see the LAR instruction.

Words

1

Cycles

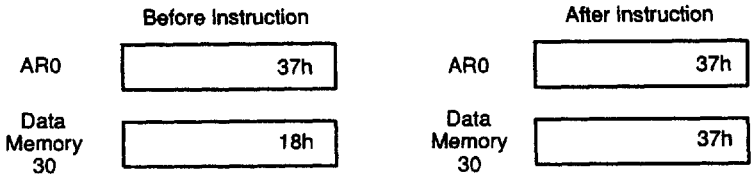
1

Example 1

```
SAR  AR0,DAT30     ;(DP = 0)
```

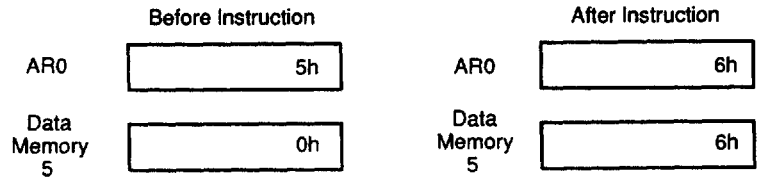
or

```
SAR  AR0,*         ;If current auxiliary register contains 30
```



Example 2

```
LARP AR0
SAR  AR0,*+
```



Special problems arise when SAR is used to store the current auxiliary register with indirect addressing if autoincrement/decrement is used.

```
LARP    AR0
LARK    AR0,10
SAR     AR0,*+ or SAR AR0,*-
```

In this case, SAR AR0,*+ stores the value 11 in location 10. SAR AR0,*– stores the value 9 in location 10.

Syntax

[*label*] SOVM

Operands

None

Operation

(PC) + 1 → PC
1 → overflow mode (OVM) status bit
Affects OVM.

Encoding

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	1	1	1	1	1	1	1	0	0	0	1	0	1	1

Description

The OVM status bit is set to logic 1, which enables the overflow (saturation) mode. If an overflow occurs with OVM set, the overflow flag OV is set, and the accumulator is set to the largest representable 32-bit positive (7FFFFFFFh) or negative (80000000h) number, according to the direction of overflow. OVM may also be loaded by the LST and ROVM instructions. (See the ROVM instruction for further information.)

Words

1

Cycles

1

Example

```
SOVM    ;The overflow mode bit OVM is set, enabling the
        ;overflow mode on any subsequent arithmetic
        ;operations
```

Syntax [*label*] SPAC

Operands None

Operation (PC) + 1 → PC
(ACC) – (P register) → ACC
Affects OV; affected by OVM.

Encoding

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	1	1	1	1	1	1	1	0	0	1	0	0	0	0

Description The contents of the P register are subtracted from the contents of the accumulator. The result is stored in the accumulator. Note that the P register is always sign-extended.

Words 1

Cycles 1

Example

```
SPAC
```

	Before Instruction		After Instruction
P	24h	P	24h
ACC	3Ch	ACC	18h

Syntax

Direct: [*label*] SST *dma*
Indirect: [*label*] SST{*|* * + | *–} [,*next ARP*]

Operands

0 ≤ dma ≤ 15 (TMS320C10)
0 ≤ dma ≤ 127 (TMS320C14/C15/C16/C17)
ARP = 0 or 1

Operation

(PC) + 1 → PC
(status register) → specified dma (page 1 only in direct addressing)

Encoding

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Direct:	0	1	1	1	1	1	0	0	0	Data Memory Address						
Indirect:	0	1	1	1	1	1	0	0	1	See Section 4.1						

Description

The status bits are saved into the specified data memory address (only page 1 if direct memory addressing is used).

In the direct addressing mode, the status register is always stored in page 1, regardless of the value of the DP register. The processor automatically forces the page to be 1, and the specific location within that page is defined in the instruction. Note that the DP register is not physically modified. This allows storage of the DP register in the data memory on interrupts, etc., in the direct addressing mode without having a change of the DP. In the indirect addressing mode, the data memory address is obtained from the auxiliary register selected. (See the LST instruction for more information.)

The SST instruction can be used to store the status bits after interrupts and subroutine calls. These status bits include the OV (overflow flag) bit, OVM (overflow mode), INTM (interrupt mode), ARP (auxiliary register pointer), and DP (data memory page pointer). The status bits are stored in the data memory word as follows:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OV	OVM	INTM	1	1	1	1	ARP	1	1	1	1	1	1	X	DP

Note: X = reserved

Words

1

Cycles

1

Example

```
SST  DAT1      ;(DP = Don't care)
```

or

```
SST  *,1       ;If current auxiliary register contains 1
```

	Before Instruction	After Instruction
Status Register	5EFEh	5EFEh
Data Memory 1	0Ah	5EFEh

Syntax

Direct: [*label*] SUB *dma* [,*shift*]
Indirect: [*label*] SUB {*|* * + | *–} [,*shift* [,*next ARP*]]

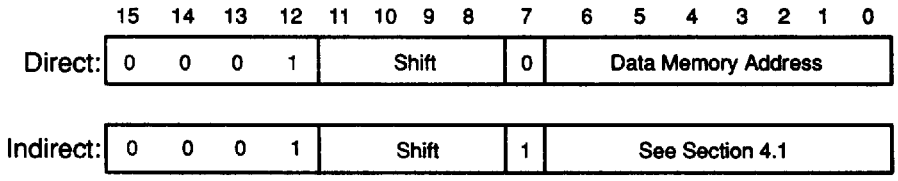
Operands

0 ≤ dma ≤ 127
ARP = 0 or 1
0 ≤ shift ≤ 15 (defaults to 0)

Operation

(PC) + 1 → PC
(ACC) – [(dma) × 2^{shift}] → ACC
Affects OV; affected by OVM.

Encoding



Description

The contents of the addressed data memory location are left-shifted and subtracted from the accumulator. During shifting, the low-order bits are zero-filled. The high-order bit is sign extended. The result is stored in the accumulator.

Words

1

Cycles

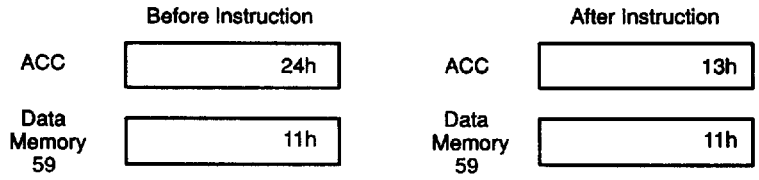
1

Example

```
SUB  DAT59          ;(DP = 0)
```

or

```
SUB  *              ;If current auxiliary register contains 59
```



Syntax

Direct: [*label*] SUBC *dma*
Indirect: [*label*] SUBC {*|* + *–} [,*next ARP*]

Operands

$0 \leq dma \leq 127$
ARP = 0 or 1

Operation

$(PC) + 1 \rightarrow PC$
$(ACC) - [(dma) \times 2^{15}] \rightarrow$ ALU output
If ALU output ≥ 0:
Then (ALU output) $\times 2 + 1 \rightarrow$ ACC;
Else (ACC) $\times 2 \rightarrow$ ACC.
Affects OV but *not* affected by OVM (no saturation).

Encoding

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Direct:	0	1	1	0	0	1	0	0	0	Data Memory Address						
Indirect:	0	1	1	0	0	1	0	0	1	See Section 4.1						

Description

The SUBC instruction performs conditional subtraction, which may be used for division. The 16-bit dividend is placed in the low accumulator, and the high accumulator is zeroed. The divisor is in data memory. SUBC is executed 16 times for 16-bit division. After completion of the last SUBC, the quotient of the division is in the lower-order 16-bit field of the accumulator, and the remainder is in the high-order 16 bits of the accumulator. SUBC assumes that the divisor and the dividend are both positive

If the 16-bit dividend contains less than 16 significant bits, the dividend may be placed in the accumulator, left-shifted by the number of leading nonsignificant zeros. The number of executions of SUBC is reduced from 16 by that number. However, at least one leading zero must always be present because both operands of the SUBC instruction must be positive. Note that the next instruction after SUBC cannot use the accumulator.

The SUBC instruction affects OV but is *not* affected by OVM. Therefore, the accumulator does not saturate upon positive or negative overflows when this instruction executes.

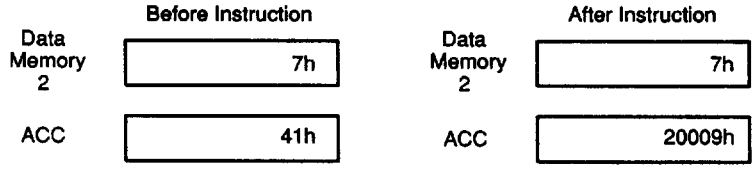
The above description is for 16-bit integer division. SUBC can also be used in fixed-point division.

Words 1

Cycles 1

Example

```
     LARP    AR0
     LARK    AR0,15
DIV  SUBC    DAT2      ;(DP = 0)
     BANZ    DIV
```



The results above show the execution of all the instructions in the code example.

Syntax

Direct: [*label*] SUBH *dma*
Indirect: [*label*] SUBH {*|* + | *–} [,*next ARP*]

Operands

$0 \leq dma \leq 127$
ARP = 0 or 1

Operation

(PC) + 1 → PC
(ACC) – [(dma) × 2^{16}] → ACC
Affects OV; affected by OVM.

Encoding

	15	14	13	12	11	10	9	8	7	6 5 4 3 2 1 0
Direct:	0	1	1	0	0	0	1	0	0	Data Memory Address
Indirect:	0	1	1	0	0	0	1	0	1	See Section 4.1

Description

The contents of the addressed data memory location are subtracted from the upper 16 bits of the accumulator. The 16 low-order bits of the accumulator are unaffected. The result is stored in the accumulator.

The SUBH instruction can be used for performing 32-bit arithmetic.

Words

1

Cycles

1

Example

```
SUBH    DAT33    ;(DP = 0)
```

or

```
SUBH    *        ;If current auxiliary register contains 33
```

	Before Instruction		After Instruction	
Data Memory 33	4h		Data Memory 33	4h
ACC	0A0013h		ACC	60013h

Syntax

Direct: [*label*] SUBS *dma*
Indirect: [*label*] SUBS {*|* + | *–} [,*next ARP*]

Operands

0 ≤ dma ≤ 127
ARP = 0 or 1

Operation

(PC) + 1 → PC
(ACC) – (dma) → ACC
Affects OV; affected by OVM.

Encoding

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Direct:	0	1	1	0	0	0	1	1	0	Data Memory Address						
Indirect:	0	1	1	0	0	0	1	1	1	See Section 4.1						

Description

The contents of the addressed data memory location are subtracted from the accumulator with sign extension suppressed. The data is treated as a 16-bit unsigned number, rather than a 2s-complement number. The accumulator behaves as a signed number.

Words

1

Cycles

1

Example

```
SUBS    DAT2     ;(DP = 0)
```

or

```
SUBS    *        ;If current auxiliary register contains 2
```

	Before Instruction		After Instruction	
Data Memory 2	0F003h		Data Memory 2	0F003h
ACC	0F105h		ACC	102h

Syntax

Direct: [*label*] TBLR *dma*
Indirect: [*label*] TBLR {*|* + *–*} [,*next ARP*]

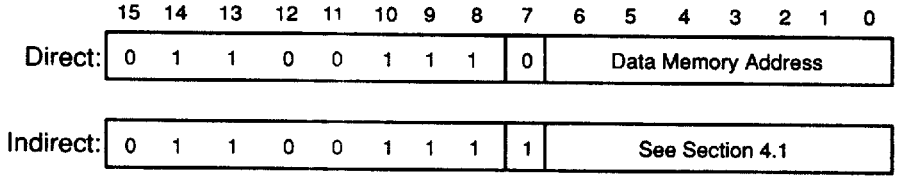
Operands

$0 \leq dma \leq 127$
ARP = 0 or 1

Operation

(PC) + 1 → TOS
(ACC(11–0)) → PC
(pma) → dma
Modify AR(ARP) and ARP as specified
(TOS) → PC

Encoding



Description

The TBLR instruction transfers a word from a location in program memory to a data memory location specified by the instruction. The program memory address is defined by the low-order 12 bits of the accumulator. For this operation, a read from program memory is performed, followed by a write to data memory. The contents of the lowest stack location are lost when TBLR is used.

The TBLR instruction is useful for reading coefficients that have have been stored in program ROM, or for reading time-dependent data stored in RAM.

Words

1

Cycles

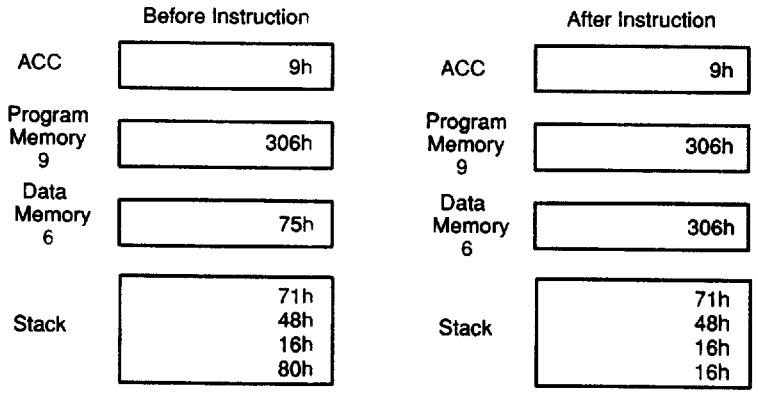
3

Example

```
TBLR    DAT6      ;(DP = 0)
```

or

```
TBLR    *         ;If current auxiliary register contains 6
```



Syntax

Direct: [*label*] TBLW *dma*
Indirect: [*label*] TBLW {*|* + | *–} [,*next ARP*]

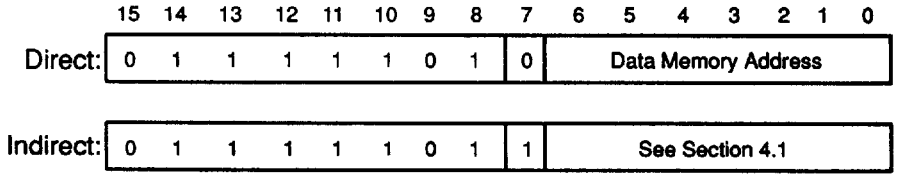
Operands

0 ≤ dma ≤ 127
ARP = 0 or 1

Operation

(PC) + 1 → TOS
(ACC(11–0)) → PC
(dma) → pma
Modify AR(ARP) and ARP as specified
(TOS) → PC

Encoding



Description

The TBLW instruction transfers a word from data memory to program memory. The data memory address is specified by the instruction, and the program memory address is specified by the lower 12 bits of the accumulator. A read from data memory is followed by a write to program memory to complete the instruction. The contents of the lowest stack location are lost when TBLW is used.

Note that the TBLW and OUT instructions use the same external signals and thus are undistinguishable during a write to program memory addresses 0 through 7. On the 'C14, this situation occurs on the highest 8 words in program memory by driving the upper address lines high instead of low.

Words

1

Cycles

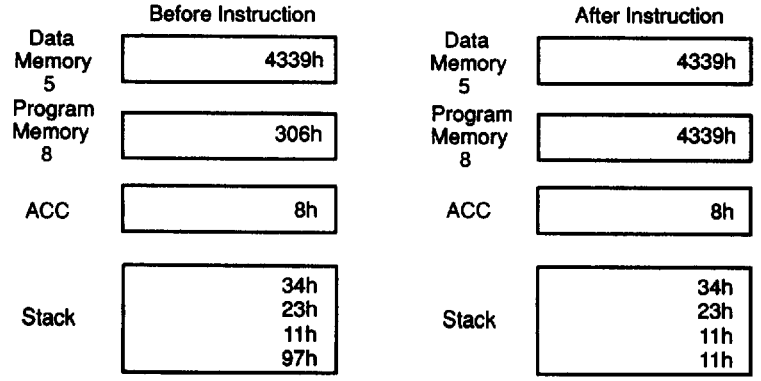
3

Example

```
TBLW    DAT5      ;(DP = 0)
```

or

```
TBLW    *         ;If current auxiliary register contains 5
```



Syntax

Direct: [*label*] XOR *dma*
Indirect: [*label*] XOR {*|* * + | *–} [,*next ARP*]

Operands

0 ≤ dma ≤ 127
ARP = 0 or 1

Operation

(PC) + 1 → PC
(ACC(15–0)).XOR.dma → ACC(15–0)
(ACC(31–16)) → ACC(31–16)

Encoding

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Direct:	0	1	1	1	1	0	0	0	0	Data Memory Address						
Indirect:	0	1	1	1	1	0	0	0	1	See Section 4.1						

Description

The low half of the accumulator is exclusive-ORed with the contents of the addressed data memory location. The upper half of the accumulator is not affected by this instruction.

The XOR instruction is useful for toggling or setting bits of a word for high-speed control. In addition, the 1s complement of a word can be found by exclusive-ORing it with all ones.

Words

1

Cycles

1

Example

```
XOR     DAT127   ;(DP = 0)
```

or

```
XOR *            ;If current auxiliary register contains 127
```

	Before Instruction	After Instruction
Data Memory 127	0F0F0h	0F0F0h
ACC	12345678h	1234A688h

Syntax [*label*] ZAC

Operands None

Operation (PC) + 1 → PC
0 → ACC

Encoding

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	1	1	1	1	1	1	1	0	0	0	1	0	0	1

Description The contents of the accumulator are replaced with zero.

Words 1

Cycles 1

Example

```
ZAC
```

	Before Instruction		After Instruction
ACC	0A5A5A5A5h	ACC	0h

Syntax

Direct: [*label*] ZALH *dma*
Indirect: [*label*] ZALH {*|* * + | *–} [,*next ARP*]

Operands

0 ≤ dma ≤ 127
ARP = 0 or 1

Operation

(PC) + 1 → PC
0 → ACC(15–0)
(dma) → ACC(31–16)

Encoding

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Direct:	0	1	1	0	0	1	0	1	0	Data Memory Address						
Indirect:	0	1	1	0	0	1	0	1	1	See Section 4.1						

Description

ZALH loads a data memory value into the high-order half of the accumulator. The low-order bits of the accumulator are zeroed.

ZALH is useful for 32-bit arithmetic operations.

Words

1

Cycles

1

Example

```
ZALH DAT3      ;(DP = 0)
```

or

```
ZALH *         ;If current auxiliary register contains 3
```

	Before Instruction	After Instruction
Data Memory 3	3F01h	3F01h
ACC	77FFFFh	3F010000h

Syntax

Direct: [*label*] ZALS *dma*
Indirect: [*label*] ZALS {*|* + | *–} [,*next ARP*]

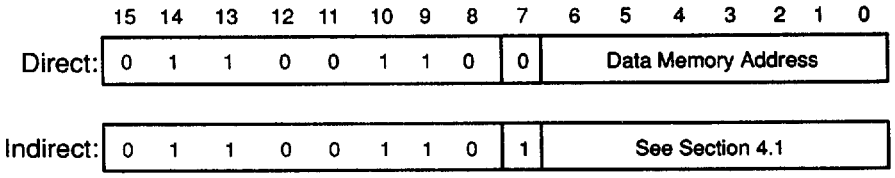
Operands

0 ≤ dma ≤ 127
ARP = 0 or 1

Operands

(PC) + 1 → PC
0 → ACC(31–16)
(dma) → ACC(15–0)

Encoding



Description

The contents of the addressed data memory location are loaded into the 16 low-order bits of the accumulator. The upper half of the accumulator is zeroed. The data is treated as a 16-bit unsigned number rather than as a 2s-complement number. Therefore, there is no sign extension with this instruction.

ZALS is useful for 32-bit arithmetic operations.

Words

1

Cycles

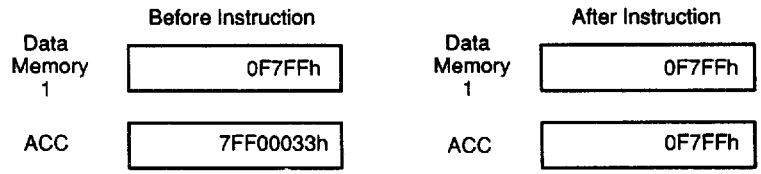
1

Example

```
ZALS    DAT1     ;(DP = 0)
```

or

```
ZALS    *        ;If current auxiliary register contains 1
```



Chapter 5

Software Applications

This chapter explains how to use key features of the software-related processor and instruction set and gives assembly language coding examples. TMS320C1x instructions are tailored to digital signal processing tasks, providing a single-cycle multiply, scaling, convolution, and overflow management. They support logical and arithmetic operations.

More information about specific applications can be found in the book, *Digital Signal Processing Applications with the TMS320 Family, Volume 1* (literature number SPRA012A). The DSP software library contains the major DSP routines and application algorithms presented in the applications book. The TMS320 DSP bulletin board service provides access to code updates and new application reports as they become available. See Appendix E for information about the software library and bulletin board.

Note:

Throughout this book, 'C14 designates all devices with a 14 suffix (C14/E14/P14), 'C15 all devices with a 15 suffix (C15/E15/LC15/P15), and 'C17 all devices with a 17 suffix (C17/E17/LC17/P17); unless otherwise noted.

Major topics discussed in this chapter are

5.1 Processor Initialization

The processor must be initialized before it can execute a digital signal processing algorithm. Generally, initialization takes place anytime the processor is reset. This section describes how to configure the TMS320C1x devices after reset and provides code for processor initialization.

When hardware reset is activated by applying a low level to the $\overline{RS}$ (reset) input for a minimum of five cycles, the TMS320C1x terminates program execution and forces the program counter (PC) to zero. After reset, program memory location 0 normally contains a B (branch) instruction to direct program execution to the system initialization routine. The hardware reset also initializes various registers and status bits.

5.1.1 Reset ($\overline{RS}$)

After reset, the processor should be initialized through software. The initialization routine sets up operational modes, memory pointers, interrupts, and the remaining functions necessary to meet system requirements.

5.1.2 TMS320C10 and TMS320C15 Initialization

To configure the TMS320C10 and TMS320C15 processor after reset, the following internal functions should be initialized:

- Interrupt structure (INTM)
- Overflow mode control (OVM)
- Auxiliary registers and auxiliary register pointer (ARP)
- Data memory page pointer (DP)

Note that the INTM (interrupt mode) bit, OVM (overflow mode) bit, auxiliary register pointer (ARP), and data memory page pointer (DP) are not initialized by reset.

Example 5–1 shows coding for initializing the TMS320C10 and the TMS320C15 to the following machine state, and also the coding for the initialization performed during the hardware reset:

- Interrupt enabled (INTM)
- Overflow mode (OVM) disabled
- Data memory page pointer (DP) set to zero
- Auxiliary register pointer (ARP) set to zero
- Internal memory filled with zeros

Example 5–1. TMS320C10 and TMS320C15 Processor Initialization

```
        .title          'PROCESSOR INITIALIZATION'
        .def    RESET,INT
        .ref    ISR
*
* PROCESSOR INITIALIZATION.
* RESET AND INTERRUPT VECTOR SPECIFICATION.
*
        .text
RESET   B       INIT        ;RS BEGINS PROCESSING HERE
INT     B       ISR         ;INT BEGINS PROCESSING HERE
*
* THE BRANCH INSTRUCTION AT PROGRAM MEMORY LOCATION 0
* DIRECTS EXECUTION TO BEGIN HERE FOR RESET PROCESSING
* THAT INITIALIZES THE PROCESSOR. WHEN RESET IS APPLIED,
* THE FOLLOWING CONDITIONS ARE ESTABLISHED FOR THE STATUS
* REGISTER:
*
*       OV  OVM INTM    12  11  10  9   ARP 7   6   5   4   3   2   DP
*ST:    0   X   1       1   1   1   1   X   1   1   1   1   1   1   X
*
INIT    ROVM            ;DISABLE OVERFLOW MODE
        LDPK    0       ;POINT DP TO DATA PAGE 0
        LARK    0,255   ;SET LOOP COUNT FOR DATA MEM INIT
                        ;TO 143 FOR 32010 AND 255 FOR
                        ;320C15/17
*
* INTERNAL DATA MEMORY INITIALIZATION.
*
        ZAC             ;CLEAR THE ACCUMULATOR
        LARP    0       ;USE AR0 FOR POINTER AND LOOP CONTROL
LOOP    SACL *          ;CLEAR DATA MEMORY BANZ LOOP
        BANZ    LOOP    ;CHECK IF DONE AND DECREMENT AR0
*
* THE PROCESSOR IS INITIALIZED. THE REMAINING APPLICATION-
* DEPENDENT PART OF THE SYSTEM SHOULD NOW BE INITIALIZED.
*
        EINT            ;ENABLE ALL INTERRUPTS
```

5.1.3 TMS320C14 Initialization

It is necessary to initialize the processor before executing a DSP algorithm. Initialization is normally done following a reset so that the processor and peripherals meet system requirements. In initializing the TMS320C14, the system requirements of the following functions should be considered:

- Auxiliary register pointer
- Data memory page pointer
- Overflow mode
- Interrupt structure
- Bit I/O
- Serial port
- Watchdog timer
- General-purpose timer
- Event manager (compare/capture)

Note that although a hardware reset forces the program counter to zero and initializes various registers and status bits, the overflow mode bit (OVM), inter-

rupt mode bit (INTM), auxiliary register pointer (ARP), and data memory page pointer bit (DP) must be initialized by the programmer.

As described in subsection 3.8.1, the on-chip peripherals are accessed with a bank address and a port address. Table 5–1 shows how the I/O registers are mapped.

Table 5–1. I/O Register Map

Port	BANK0	BANK1	BANK2	BANK3	BANK4	BANK5	BANK6	BANK7	BANKFFFF
0	IOP	WDT	TMR1	CMPR0	ACT0	SCON	FIFO0		EXT. I/O
1	DDR	WPER	TPR1	CMPR1	ACT1	SSET	FIFO1		EXT. I/O
2	BSET	WTPL4	TMR2	CMPR2	ACT2	SCLR	FIFO2	SMAT	EXT. I/O
3	BCLR	SYSCON	TPR2	CMPR2	ACT3	TBR	FIFO3	TSR	EXT. I/O
4	IF		TCON	CMPR4	ACT4	RBR	CCON	RSR	EXT. I/O
5	IM			CMPR5	ACT5	SBRG	CCLR	STMR	EXT. I/O
6	FCLR								EXT. I/O
7	BSR	BSR	BSR	BSR	BSR	BSR	BSR	BSR	BSR

The assembly language code presented in Example 5–2 and in Example 5–3 initializes the TMS320C14 to the following machine state:

- Overflow mode disabled
- Data memory page pointer set to zero
- Auxiliary register pointer set to zero
- Internal memory filled with zeros
- Bit I/O
  - Bits 15 through 8 configured as inputs
  - Bits 7 through 0 configured as outputs
- Serial port configured as:
  - Asynchronous mode
  - Seven data bits
  - Even parity
  - 19.2-kbps bit rate
- Watchdog timer set to time out once every 1 ms.
- Use the compare subsystem and set compare data to a specific pattern. If match occurs, output pin is toggled appropriately as defined by action register.
- Interrupt enabled
- Timer 1 and 2 used by the compare subsystem are initialized to 400 µs.

The header file in Example 5–2 is included in each of the TMS320C14 example program listings in this chapter. It defines constants used for symbolic access to registers and bit field maps of control registers. The following example listings refer to the header file simply as C14INC.

Example 5–2. Header File Constants(TMS320C14)

```
*
                    .title 'HEADER FILE CONSTANTS'
*
*                   FILE NAME: C14INC
*
*THIS HEADER FILE DEFINES CONSTANTS FOR SYMBOLIC ACCESS
*TO REGISTERS AND BIT MAP FIELDS OF CONTROL REGISTERS
*
*
*DEFINE CONSTANTS FOR BITS
*
BIT0                .set    0000000000000001B
BIT1                .set    0000000000000010B
BIT2                .set    0000000000000100B
BIT3                .set    0000000000001000B
BIT4                .set    0000000000010000B
BIT5                .set    0000000000100000B
BIT6                .set    0000000001000000B
BIT7                .set    0000000010000000B
BIT8                .set    0000000100000000B
BIT9                .set    0000001000000000B
BIT10               .set    0000010000000000B
BIT11               .set    0000100000000000B
BIT12               .set    0001000000000000B
BIT13               .set    0010000000000000B
BIT14               .set    0100000000000000B
BIT15               .set    1000000000000000B
*
MASK0               .set    1111111111111110B
MASK1               .set    1111111111111101B
MASK2               .set    1111111111111011B
MASK3               .set    1111111111110111B
MASK4               .set    1111111111101111B
MASK5               .set    1111111111011111B
MASK6               .set    1111111110111111B
MASK7               .set    1111111101111111B
MASK8               .set    1111111011111111B
MASK9               .set    1111110111111111B
MASK10              .set    1111101111111111B
MASK11              .set    1111011111111111B
MASK12              .set    1110111111111111B
MASK13              .set    1101111111111111B
MASK14              .set    1011111111111111B
MASK15              .set    0111111111111111B
```

```
*
BitIOBank           .set    0
InterruptBank       .set    0
WDTBank             .set    1
SYSBank             .set    1
TimerBank           .set    2
CompareBank         .set    3
ActionBank          .set    4
SerialPort          .set    5
CaptureBank         .set    6
SerialPortRegs      .set    7
*
ExternalIO          .set    0FFFFH
*
BSR                 .set    7
*
*      DEFINE REGISTERS IN BitIOBank
*
IOP                 .set    0
DDR                 .set    1
BSET                .set    2
BCLR                .set    3
*
*      DEFINE REGISTERS FOR InterruptBank
*
IF                  .set    4
IM                  .set    5
FCLR                .set    6
*
*      DEFINE REGISTERS IN WDTBank
*
WDT                 .set    0
TMR4                .set    0
WPER                .set    1
TPR4                .set    1
WTPL                .set    2
*
*      DEFINE REGISTER IN SYSBank
*
SYSCON              .set    3
*
*      DEFINE REGISTERS IN TimerBank
*
TMR1                .set    0
TPR1                .set    1
TMR2                .set    2
TPR2                .set    3
TCON                .set    4
*
*      DEFINE REGISTERS IN CompareBank
*
CMPR0               .set    0
CMPR1               .set    1
CMPR2               .set    2
CMPR3               .set    3
CMPR4               .set    4
CMPR5               .set    5
*
*      DEFINE REGISTERS IN ActionBank
*
ACT0                .set    0
ACT1                .set    1
ACT2                .set    2
ACT3                .set    3
ACT4                .set    4
ACT5                .set    5
*
```

```
*     DEFINE REGISTERS IN SerialPort
*
SCON            .set    0
SSET            .set    1
SCLR            .set    2
TBR             .set    3
RBR             .set    4
SBRG            .set    5
*
*     DEFINE REGISTERS IN CaptureBank
*
FIFO0           .set    0
FIFO1           .set    1
FIFO2           .set    2
FIFO3           .set    3
CCON            .set    4
CCLR            .set    5
*
*     DEFINE SerialPortRegs
*
SMAT            .set    2
TSR             .set    3
RSR             .set    4
STMR            .set    5
*
*     DEFINE BIT MAP FOR SYSCON
*
MCMPbit         .set    BIT0
*
*     DEFINE BIT MAP FOR STATUS REGISTER
*
_OV             .set    BIT15
_OVM            .set    BIT14
_INTM           .set    BIT13
_ARP            .set    BIT8
_DP             .set    BIT0
*
*     DEFINE BIT MAP FOR INTERRUPT FLAG REGISTER
*
_NMI            .set    BIT15
_INT            .set    BIT14
_STMRINT        .set    BIT12
_CAPINT3        .set    BIT11
_CAPINT2        .set    BIT10
_CAPINT1        .set    BIT9
_CAPINT0        .set    BIT8
_CMPINT1        .set    BIT7
_CMPINT0        .set    BIT6
_TIMINT2        .set    BIT5
_TIMINT1        .set    BIT4
_RXINT          .set    BIT3
_TXINT          .set    BIT2
_WDTINT         .set    BIT1
_IOPINT         .set    BIT0
*
```

```
*     DEFINE SHIFTS FOR INTERRUPT BIT MAP
*
NMI_BIT          .set    15
INT_BIT          .set    14
STMRINT_BIT      .set    12
CAPINT3_BIT      .set    11
CAPINT2_BIT      .set    10
CAPINT1_BIT      .set    9
CAPINT0_BIT      .set    8
CAPINT1_BIT      .set    7
CMPINT0_BIT      .set    6
TIMINT2_BIT      .set    5
TIMINT1_BIT      .set    4
RXINT_BIT        .set    3
TXINT_BIT        .set    2
WDTINT_BIT       .set    1
IOPINT_BIT       .set    0
*
*      DEFINE BIT MAP FOR TCON BIT
*
CAPIntOnFirst    .set    BIT15       ;15
CAPIntOnThird    .set    0
*
CaptureEnable    .set    0           ;14
CaptureDisable   .set    BIT14
*
CAPTMR2Select    .set    BIT13       ;13
CAPTMR1Select    .set    0
*
CMP5Enable       .set    Bit12       ;12
CAP3Enable       .set    0
*
CMP4Enable       .set    BIT11       ;11
CAP2Enable       .set    0
*
CMPEnable        .set    BIT10       ;10
*
CMPTMR2Select    .set    BIT9        ;9
CMPTMR1Select    .set    0
*
CMPPWMMode       .set    BIT8        ;8
CMPNormalMode    .set    0
*
TMR2Enable       .set    BIT6        ;7,6 TIMER 2 MODE FIELD
*
TMR2Internal     .set    0           ;5,4 TIMER 2 SOURCE
TMR2External     .set    BIT4+BIT5
TMR2fromTMR1     .set    BIT5
*
TMR1Enable       .set    BIT1        ;2,1 TIMER 1 MODE FIELD
*
TMR1Internal     .set    0           ;0 TIMER 1 SOURCE
TMR1External     .set    1
*
*
*     TIMER 1 AND 2 MODES
TMRStop          .set    0
TMRby4           .set    1
TMRby16          .set    2
TMRby1           .set    3
*
```

```
*     BIT MAP FOR ACTION REGISTER
*
*    ACTION CODES
*
CMPNoaction     .set    0
CMPReset        .set    1
CMPSet          .set    2
CMPToggle       .set    3
*
*    COMPARE OUTPUT ACTION FIELDS
*
CMP0Act         .set    BIT14
CMP1Act         .set    BIT12
CMP2Act         .set    BIT10
CMP3Act         .set    BIT8
CMP4Act         .set    BIT6
CMP5Act         .set    BIT4
*
*    COMPARE INTERRUPT ENABLES
*
CMPINT1Enable   .set    BIT3
CMPINT2Enable   .set    BIT2
ActionDefault   .set    BIT1+BIT0
*
*     BIT MAP FOR CCON
*
*     CAPTURE CONTROL BIT FIELD DEFINITIONS
*
*     FIFO STATUS BITS
*
FIFOFull        .set    8
FIFONotEmpty    .set    4
*
*    CAPTURE MODE DEFINITIONS
*
CAPDisable      .set    0
PosEdgeDetect   .set    1
NegEdgeDetect   .set    2
PosNegDetect    .set    3
*
*    CAPTURE CONTROL FIELDS
*
CAP3Mode        .set    BIT12
CAP2Mode        .set    BIT8
CAP1Mode        .set    BIT4
CAP0Mode        .set    BIT0
*
*
*     BIT MAP FOR SCON
                        SCON BIT
*
*
SynchMode       .set    BIT0        ;0
AsynchMode      .set    0
*
ParityEnable    .set    BIT1        ;1
ParityDisble    .set    0
*
OddParity       .set    BIT2        ;2
EvenParity      .set    0
*
ParityError     .set    BIT3        ;3
*
FERR            .set    BIT4        ;4
*
```

```
*     NUMBER OF DATA BITS SELECT FIELD
*
SixDataBits     .set    0           ;5,6
SevenDataBits   .set    BIT5
EightDataBits   .set    BIT6
NineDataBits    .set    BIT5+BIT6
*
RxOverflow      .set    BIT7        ;7
*
RxFull          .set    BIT8        ;8
*
*     CODEC MODE FIELD
*     SEE TABLE 3-19 FOR DEFINITIONS
*
CodecMode       .set    BIT9        ;9,10,11,12,13
*
CodecEnable     .set    BIT14       ;14
*
TxEmpty         .set    BIT15       ;15
*
*     EQUATES FOR BAUD RATE WITH 25.6 MHZ CLOCK
*
SBRG19200       .set    20
SBRG9600        .set    40
SBRG4800        .set    82
SBRG2400        .set    166
SBRG1200        .set    332
SBRG300         .set    1332
*
*      EQUATES FOR WDT RESET
*
WDTmagic1       .set    0ABCDh
WDTmagic2       .set    2345h
```

Example 5–3. TMS320C14 Processor Initialization

```
*
*     INITIALIZATION FOR THE C14/E14
*
              .title 'C14/E14 PROCESSOR INITIALIZATION'
*
              .include   "C14INC"    ;INCLUDE HEADER FILE
*
WDT1mS        .set    782             ;COUNTS FOR1 MS WITH MAX CLOCK RATE
TMR2Period    .set    3125            ;PERIOD FORTIMER2 (400 MICROSEC)
IOPortData    .set    0A5h
InterruptMask .set    0FFFCh          ;MASK FOR IM   TO ENABLE WDT
*                                     ;AND IOP INTERRUPTS
*
SerialPortMode .set   AsynchMode+SevenDataBits+ParityEnable+EvenParity
CaptureMode   .set    CAPIntOnFirst+CAPTMR2Select
CompareMode   .set    CMP5Enable+CMPEnable
Timer2Mode    .set    TMR2Enable*TMRby1+TMR2Internal
Timer1Mode    .set    TMR1Enable*TMRStop
ActionMode    .set    CMP5Act*CMPToggle+ActionDefault
CaptureControl .set   PosNegDetect*CAP2Mode
*
              .bss    ONE,1
              .bss    TMP,1
              .bss    BANK,1
              .bss    REG,1
              .bss    VALUE,1
              .bss    WDTVAL1,1
              .bss    WDTVAL2,1
              .bss    INTMSK,1
*
              .ref    ISR
*
```

```
* RESET AND INTERRUPT BRANCH INSTRUCTIONS
*
              .text
*
              B       INIT
              B       ISR
*
* PLACE INIT TABLE IN INITIALIZED DATA SPACE
              .data
*
InitTable:
              .word   WDT1mS
              .word   WDTmagic1
              .word   WDTmagic2
              .word   SerialPortMode ;SERIAL PORT MODE
              .word   SBRG19200
              .word   CaptureMode+CompareMode+Timer1Mode+Timer2Mode
              .word   TMR2Period
              .word   InterruptMask
              .word   ActionMode
              .word   CaptureControl
*
* BACK TO PROGRAM SPACE
*
              .text
*
INIT:
              ROVM                    ;DISABLE OVERFLOW MODE
              LDPK    0               ;POINT TO DATA PAGE 0
              LARK    AR0,255         ;AR0 = 255
*
* CLEAR DATA RAM
*
              ZAC
              LARP    AR0             ;ARP -> AR0
LOOP:
              SACL    *               ;CLEAR MEMORY POINTED TO BY AR0
              BANZ    LOOP            ;DECREMENT AR0 AND BRANCH TO LOOP
                                      ;IF AR0 <> 0
*
              LACK    1
              SACL    ONE             ;ONE = 1
*
              LT      ONE
              MPYK    InitTable
              PAC                     ;ACC = ->InitTable
                                      ;SINCE LACK ONLY REFERENCES AN 8
                                      ;BIT CONSTANT AND MPYK REFERENCES
                                      ;A 13BIT CONSTANT, MULTIPLYING BY
                                      ;1 AND THEN SAVING THE RESULT IN
                                      ;THE ACC THROUGH THE P REGISTER
                                      ;SOLVES THE PROBLEM OF GETTING A
                                      ;12 BIT ADDRESS INTO THE ACC
              TBLR    VALUE           ;VALUE = InitTable[0]
              SACL    TMP             ;TMP = ->InitTable[0]
*
              LACK    WDTBank         ;INITIALIZE THE WDT PERIOD
                                      ;REGISTER
              SACL    BANK            ;LOAD WPER WITH COUNTS FOR 1MS
              OUT     BANK,BSR        ;TIMEOUT
              OUT     VALUE,WPER
*
              LAC     TMP
              ADD     ONE
              SACL    TMP             ;TMP  = ->InitTable[1]
              TBLR    WDTVAL1         ;INITIALIZE WDT MAGIC VALUE 1
*
```

```
                LAC     TMP
                ADD     ONE
                SACL    TMP             ;TMP  = ->InitTable[2]
                SACL    WDTVAL2         ;INITIALIZEWDT MAGIC VALUE 2
*
                LAC     TMP
                ADD     ONE
                SACL    TMP             ;TMP  = ->InitTable[3]
                TBLR    VALUE           ;INITIALIZE SCON REGISTER
                LACK    SerialPort      ;ASYNCHRONOUS MODE
                SACL    BANK            ;7 DATA BITS
                OUT     BANK,BSR        ;EVEN PARITY
                OUT     VALUE,SCON
*
                LAC     TMP
                ADD     ONE
                SACL    TMP             ;TMP = ->InitTable[4]
                TBLR    VALUE           ;SET BAUD RATE GENERATOR FOR
                                        ;19200 BAUD
                OUT     VALUE,SBRG
*
                LAC     TMP
                ADD     ONE
                SACL    TMP             ;TMP = ->InitTable[5]
                TBLR    VALUE           ;INITIALIZE TCON
                LACK    TimerBank       ;ENABLE CAPTURE, INTERRUPT ON
                                        ;FIRST INPUT
                SACL    BANK            ;COMPARE 5: TOGGLE MODE
                OUT     BANK,BSR        ;TIMER 1 STOP
                OUT     VALUE,TCON      ;TIMER 2 DIVIDE BY 1, INTERNAL
                                        ;CLOCK
*
                LAC     TMP
                ADD     ONE
                SACL    TMP             ;TMP = ->InitTable[6]
                TBLR    VALUE           ;INITIALIZE TMR2 PERIOD
                OUT     VALUE,TPR2      ;LOAD WITH COUNTS FOR 400
                                        ;MICROSEC PERIOD WITH 25.6MHZ
                                        ;CLOCK INPUT
*
                LAC     TMP
                ADD     ONE
                SACL    TMP             ;TMP  = ->InitTable[7]
                TBLR    VALUE           ;INITIALIZE INTERRUPT MASK
                LACK    InterruptBank   ;ENABLE WDT AND IOP
                                        ;INTERRUPTS
                SACL    BANK
                OUT     BANK,BSR
                OUT     VALUE,IM
*
                LAC     TMP
                ADD     ONE
                SACL    TMP             ;TMP  = ->InitTable[8]
                TBLR    VALUE           ;INITIALIZE ACTION MODE FOR CMP5
                LACK    ActionBank      ;TOGGLE COMPARE OUTPUT 5
                SACL    BANK
                OUT     BANK,BSR
                OUT     VALUE,ACT5
*
                LAC     TMP
                ADD     ONE
                SACL    TMP             ;TMP = ->InitTable[9]
                TBLR    VALUE           ;INITIALIZE CAPTURE CONTROL
                LACK    CaptureBank     ;DETECT RISING OR FALLING EDGES ON CAP
                SACL    BANK
                OUT     BANK,BSR
                OUT     VALUE,CCON
*
*       INITIALIZE DDR
```

```
*
*       CONFIGURE BITS 0-7 OF IOP AS OUTPUTS AND 8-15 OF IOP AS INPUTS
*
              LACK    BitIOBank
              SACL    BANK
              OUT     BANK,BSR
              LACK    11111111B
              SACL    VALUE
              OUT     VALUE,DDR
*
*       THIS COMPLETES DEVICE INITIALIZATION
*
              EINT                ;ENABLE INTERRUPTS
*
       END
```

5.1.4 TMS320C17 Initialization

To configure the TMS320C17 after reset, the following internal functions must be initialized:

- Interrupt structure
- Serial-port framing-pulse generation selection
- Serial-port connection
- Companding hardware
- Serial-port clock
- Auxiliary register pointer
- Data memory page pointer
- Overflow mode

Two of the I/O ports are dedicated to the serial port and companding hardware, the operation of which is determined by the 32 bits of the system control register. Table 5–2 lists the control register bits and gives brief definitions.

Table 5–2. Control Register Bit Definitions

Control Register Bit #	Definition
Port 0	
CR3–CR0	Interrupt flags
CR7–CR4	Interrupt mask bits
CR8	Port 1 configuration control
CR9	External framing enable for serial port transfers
CR10	XF external logic output flag latch
CR11	Serial port companding mode select
CR13–CR12	Companding hardware enable
CR14	A-law/μ-law conversion select
CR15	Serial clock (SCLK) control

Table 5–2. Control Register Bit Definitions (Continued)

Control Register Bit #	Definition
Port 1	
CR23–CR16	Frame counter modulus
CR27–CR24	Serial clock (SCLK) prescale control (divide ratios)
CR28	FR pulse-width control
CR30–CR29	I/O control
CR31	Reserved for future expansion (set to 0)

Example 5–4 shows coding for initializing the TMS320C17 serial-port and companding hardware for interface to a codec. The following machine state is loaded:

- Load the value 0B988h into the lower control register.This sets bit (CR8) enabling port 1 to access the upper control register. This also sets (R15) to logic 1, which configures SCLK as an input to the device. This insures safe system operation (SCLK line contention is avoided) and prevents invalid serial port timing during the initialization routine.

- The upper control register is set as follows:
  - Long FR pulse (variable data-rate selected)
  - SCLK divide ratio of 10
  - FR frequency at SCLK/256 for an 8-kHz framing pulse
  - The value 7CFEh is loaded into the upper control register

Note that the data operand of the upper control register is set at 7CFEh. This selects 2s-complement companding for the serial port and 16-bit length coprocessor mode (that is, for interface to 16-bit processors). When 2s-complement companding is used, there must be at least one instruction between an OUT instruction to the serial port transmit register and an IN instruction from the serial port receive register.

- The lower control register is then configured as follows:
  - Interrupt flags cleared
  - Active FR interrupt enabled. (The FR interrupt flag is generated independently of the enable condition to the serial port)
  - Port transfers enabled by active FR
  - Serial companding mode selected (see subsection 5.6.1)
  - Companding hardware enabled
  - μ-law conversion selected
  - SCLK selected as an output
  - The value 3888h now loaded into the lower control register.

Note that the interrupt flags are flip-flops. Writing a one to an interrupt flag clears it and sets the corresponding flag to zero; that is, a write to the flags affects the clear or reset input of the flip-flops.

Example 5–4. TMS320C17 Processor Initialization

```
*A BRANCH INSTRUCTION AT PROGRAM MEMORY LOCATION 0 DIRECTS
*PROCESSOR EXECUTION HERE. THE CONTROL REGISTER VALUES ARE
*STORED IN ROM STARTING AT LOCATION 4. THESE VALUES ARE
*THEN READ INTO RAM FOR THE OUT INSTRUCTIONS TO THE CONTROL
*REGISTER. MEMORY LOCATIONS SET1-SET3 AND ONE ARE LOCATED
*ON RAM PAGE 1. THE PROGRAM MEMORY LOCATION HAS A BRANCH TO
*THE INTERRUPT SERVICE ROUTINE.
*
       .def   RESET,INT,INIT
       .ref   ISR
*
ONE    .set   1              ;CONSTANT ONE
SET1   .set   2              ;LOWER CONTROL REGISTER
SET2   .set   3              ;UPPER CONTROL REGISTER
SET3   .set   4              ;LOWER CONTROL REGISTER
*
* PROCESSOR INITIALIZATION.
* RESET AND INTERRUPT VECTOR SPECIFICATION.
*
       text
RESET  B      INIT           ;RS BEGINS PROCESSING HERE
INT    B      ISR            ;INT BEGINS PROCESSING HERE
TABLE  .word  0B988h         ;CONTROL REGISTER DATA
       .word  7CFEh
       .word  3888h
*
INIT   DINT                  ;DISABLE INTERRUPTS
       SOVM                  ;SET OVERFLOW MODE
       LARP   0              ;USE AUXILIARY REGISTER 0
       LDPK   1              ;WORK IN RAM PAGE 1
       LACK   1              ;ACC = 1
       SACL   ONE            ;STORE 1 IN MEMORY LOCATION ONE
       LACK   TABLE          ;START AT LOCATION 4
       TBLR   SET1           ;READ VALUE 0B988h TO RAM
       ADD    ONE,0          ;INCREMENT ADDRESS
       TBLR   SET2           ;READ VALUE 1CFEh TO RAM
       ADD    ONE,0          ;INCREMENT ADDRESS
       TBLR   SET3           ;READ VALUE 3888h TO RAM
       OUT    SET1,0         ;CONFIGURE LOWER CONTROL REGISTER
       OUT    SET2,1         ;CONFIGURE UPPER CONTROL REGISTER
       OUT    SET3,0         ;CONFIGURE LOWER CONTROL REGISTER
       LDPK   0              ;RESET RAM PAGE TO 0
*
* THE PROCESSOR IS INITIALIZED. THE REST OF THE SYSTEM THAT
* IS APPLICATION-DEPENDENT SHOULD BE INITIALIZED BEFORE THE
* EINT INSTRUCTION.
*
       EINT                  ;ENABLE INTERRUPTS
```

5.2 Interrupt Management

The interrupt function allows the current process to be suspended to perform a more critical function. On the TMS320C10 and the TMS320C15, processor execution may be suspended on a high-priority basis by using the $\overline{INT}$ pin. Otherwise, a lower priority interrupt can be serviced by using a software ($\overline{BIO}$) polling technique.

Four interrupts on the TMS320C17 and 15 interrupts on TMS320C14 are maskable via the system control register. These interrupts are synchronized and multiplexed into the master interrupt circuitry and have the same priority. Software polling techniques are used to determine which input caused the interrupt when multiple interrupts are enabled.

Processing in the interrupt service routine (ISR) must assure that the processor context is saved before and during execution and restored when the routine is finished. Descriptions and examples of how to implement interrupt service routines, $\overline{BIO}$ polling, and context switching are provided in this section.

5.2.1 TMS320C10 and '15 Interrupt Service Routine

The TMS320C10 and TMS320C15 devices provide one maskable interrupt ($\overline{INT}$). Using the $\overline{INT}$ pin permits the processor's execution to be suspended at any point in the program except after a multiply instruction. The instruction following the MPY and MPYK instructions is always executed.

Interrupt processing on the TMS320C10, TMS320C15 begins as follows:

1) The EINT (enable interrupt) instruction is executed, which sets the INTM (interrupt mode) bit to 0 so that an interrupt can be received.

2) When an interrupt occurs, the INTF (interrupt flag) bit is set to 1.

As interrupt servicing begins, the following sequence occurs automatically:

1) The interrupt is acknowledged, which clears the INTF (interrupt flag) bit to 0.

2) The INTM (interrupt mode) bit is set to 1 to disable further interrupts.

3) The current PC is pushed onto the TOS (top of stack).

4) The new PC is set to 2.

During servicing of the interrupt, you commonly perform the following operations in software:

1) The interrupt service routine is executed. Program memory address 2 will have either a service routine to save the context of the machine or a branch to the interrupt service routine.The context of the machine can be saved and the source of the interrupt serviced. Then, the context is restored and the interrupts enabled prior to returning from the interrupt routine.
2) The EINT (enable interrupt) instruction is executed, which sets the INTM (interrupt mode) bit to 0.
3) The RET instruction is executed.

The hardware interrupt can be masked at critical points in the program with the DINT instruction. This sets the INTM (disable interrupt mode) bit to logic one. If an interrupt occurs while INTM equals one, the interrupt will not be serviced until the interrupts are enabled again. However, the INTF(interrupt flag) is set to one, and the interrupt is held pending. The interrupt will be serviced when the INTM bit is set to zero by executing the EINT instruction. If an interrupt is pending when an enable interrupt operation occurs, the interrupt is serviced after the execution of the instruction following the EINT instruction. This allows for a return instruction to be executed before an interrupt is acknowledged.

An interrupt-driven analog input channel can be implemented by using the technique described and shown in Example 5–5. However, multiple-level data buffering will impact system I/O overhead. Analog systems supported by TMS320C1x devices usually have information bandwidths of less than 20 kHz. The desired sample rate can be generated by dividing the CLKOUT signal from the TMS320 processor. It is advisable to provide at least a one-level data buffer to ensure the integrity of the data read by the processor. If an 8-kHz sample rate is used (for example), the system must then respond to an analog interrupt every 125 µs. The percentage of I/O overhead incurred by this arrangement can be computed by determining the number of clock cycles that the TMS320 processor spends in the interrupt routine servicing each sample and dividing by the number of clock cycles available between each sample. Example 5–5 shows a typical interrupt service routine. Note that the memory location flag (FLAG) contains a 1-bit flag to indicate that the required number of samples has been received.

Example 5–5. TMS320C10 and TMS320C15 Interrupt Service Routine

```
* THIS ROUTINE SERVICES AN EXTERNAL INTERRUPT. IT MAY BE
* LOCATED AT PROGRAM MEMORY LOCATION 2, OR A BRANCH AT
* LOCATION 2 DIRECTS PROGRAM EXECUTION HERE. THE ROUTINE
* READS DATA FROM AN EXTERNAL DEVICE (A/D CONVERTER). THE
* NUMBER OF SAMPLES OBTAINED ARE STORED IN MEMORY LOCATION
* COUNT. LIMIT IS THE NUMBER OF SAMPLES NEEDED. MEMORY
* LOCATION ONE CONTAINS THE CONSTANT 1. STATUS IS ALWAYS
* STORED ON DATA PAGE 1 WHEN USING DIRECT MEMORY ADDRESSING.
* ASSUME AR0 POINTS TO THE NEXT EMPTY LOCATION IN THE SAMPLE
* BUFFER.
*
```

```
ADC     .set 0          ;ASSIGN PA0 TO A/D CONVERTER
STATUS  .set 0          ;STATUS REGISTER STORAGE ON PAGE 1
ACCL    .set 1          ;ASSIGN LOCATION TO SAVE STATUS/ACC
ACCH    .set 2
SAMP    .set 3          ;STORE INPUT DATA HERE
COUNT   .set 4          ;COUNT # OF SAMPLES HERE
FLAG    .set 5          ;ASSIGN MEM LOCATION TO FLAG
LIMIT   .set 32         ;ASSIGN TOTAL # OF SAMPLES REQUIRED
*
        .text
ISR     SST  STATUS     ;SAVE STATUS
        LDPK 1          ;USE DATA PAGE 1
        SACL ACCL       ;SAVE ACCUMULATOR LOW
        SACH ACCH       ;SAVE ACCUMULATOR HIGH
        LARP 0          ;USE AR0
        IN   *-,ADC     ;READ FROM ADC
        LAC  COUNT      ;LOAD SAMPLE COUNTER
        ADD  ONE        ;INCREMENT
        SACL COUNT      ;STORE UPDATED COUNT
        LACK LIMIT
        SUB  COUNT      ;CHECK IF LIMIT EXCEEDED
        BGZ  OK
DONE    LACK 1
        SACL FLAG       ;YES --> SET FLAG
OK      ZALH ACCH       ;RESTORE ACCUMULATOR HIGH
        ADDS ACCL       ;RESTORE ACCUMULATOR LOW
        LST  STATUS     ;RESTORE STATUS
        EINT            ;ENABLE SUBSEQUENT INTERRUPTS
        RET
```

If the processor is using a 20-MHz clock, the number of available cycles between each sample is 625. The overhead required to service this system is 18/625 = 2.9 percent. This overhead burden can be reduced by using a FIFO (first in, first out) to buffer the data. In this case, the TMS320 device needs to be interrupted only when the buffer has filled. If a 16-level FIFO is used in the example above, this interrupt occurs every 2 ms, and the overhead burden is reduced to about 0.5 percent.

If two different kinds of devices are being serviced by the same interrupt routine, the $\overline{BIO}$ pin can be used to determine which device needs to be serviced (see subsection 5.2.4 for $\overline{BIO}$ polling).

5.2.2 TMS320C14 Interrupt Service Routine

The TMS320C14 provides a total of 15 external and internal interrupts. Two interrupts are dedicated for external sources. The remaining interrupts service the on-chip peripherals. All interrupts, except $\overline{NMI}$, are maskable through an interrupt mask register (IM). The interrupts are synchronized and multiplexed into the master interrupt circuitry and have the same priority. Software polling techniques are used to determine which input caused the interrupt.

Interrupt processing on the TMS320C14 begins as follows:

1) The EINT (enable interrupt) instruction is executed, which sets the INTM (interrupt mode) bit to 0 so that an interrupt can be received.

2) When an interrupt occurs, the IF (interrupt flag) bit is set to 1.
3) If the corresponding interrupt mask (IM) bit is zero, the CPU interrupt is generated.

As the interrupt is generated (either by an internal or external source), the following sequence occurs automatically:

1) The INTM bit is set to 1 to disable further interrupts.
2) The current PC is pushed onto the TOS (top of stack).
3) The new PC is set to 2.

During the servicing of the interrupt, you commonly perform the following operations in software:

1) Program memory address 2 will either have a service routine to save the context of the machine or branch to the interrupt service routine.
2) The interrupt service routine is executed. The context of the machine may be stored and restored later if required. The following sequence can be used to select which interrupt to service:
   a) Use software polling techniques to determine which one of the 15 flags has been set in the control register.
   b) Check for corresponding mask bits before proceeding (optional).
   c) Clear the flag (reset to 0) through the FCLR register and service the source of the flag.
3) The EINT instruction is executed, clearing the INTM bit to 0.
4) The RET instruction is executed.

Although all interrupts have the same hardware priority, you can control the polling of the interrupt flags. The ISR should clear the interrupt flag before executing the EINT instruction or enabling interrupts. Note that clearing the flag register requires writing a one to the FCLR register. Writing a zero has no effect. The interrupt service routine in Example 5–6 is for a system with five active interrupts and includes polling.

Example 5–6. TMS320C14 Interrupt Service Routine

```
* THIS IS AN EXAMPLE INTERRUPT SERVICE ROUTINE
*
* THIS ROUTINE MAY BE LOCATED AT LOCATION 2 TO BE INVOKED THROUGH
* A BRANCH LOCATED AT LOCATION 2. THIS MODULE IS DESIGNED AS A
* DISPATCHER FOR THE VARIOUS SERVICE ROUTINES THAT WOULD BE REQUIRED
* TO IMPLEMENT THE DESIRED RESPONSES TO SYSTEM INTERRUPTS.
*
        .include "C14INC"              ;INCLUDE HEADER FILE
*
        .def     ISR,ISRexit
*
        .ref     Rxisr,Txisr,WDTisr,NMIisr,INTisr,IOPisr
        .ref     ONE                   ;INITIALIZED TO 1
*
```

```
        .ref    BANK                ;THESE 7 LOCAL VARIABLES SHOULD BE
        .ref    STATUS              ;PLACED ON PAGE 1
        .ref    ACCL
        .ref    ACCH
        .ref    BankSave
        .ref    TMP
        .ref    IFimage
*
        .text
*
ISR:
*
SAVE ENVIRONMENT
*
        SST     STATUS              ;SAVE STATUS ON DATA PAGE 1
        LDPK    1                   ;DP NOW POINTS TO DATA PAGE 1
        SACL    ACCL                ;SAVE ACCUMULATOR
        SACH    ACCH
        IN      BankSave,BSR        ;SAVE BSR
*
* BRANCH TO THE APPROPRIATE INTERRUPT SERVICE ROUTINE BASED
* ON BIT MAP OF IF
*
        OUT     InterruptBank,BSR   ;GET INTERRUPT FLAGS
        IN      IFimage,IF          ;STORE IN IFimage
        LAC     ONE,15              ;TMP = 7FFFh
        SUB     ONE
        SACL    TMP
        OUT     TMP,FCLR            ;CLEAR ALL INTERRUPTS
*
* TEST FOR INTERRUPTS FROM:
*                   NMI
*                   TRANSMITTER AND RECEIVE SECTIONS
*                   OF SERIAL PORT
*                   WDT
*                   BIT I/O PORT
*                   INT PIN
*
*           IGNORE ALL OTHER INTERRUPTS
*
        LAC     ONE,NMI_BIT
        AN      IFimage
        BZ      SkipNMI             ;IF (NMI) NMIisr()
        CALL    NMIisr
SkipNMI:
        LAC     ONE,RXINT_BIT
        AND     IFimage
        BZ      SkipRx              ;IF (RECEIVE PORT INTERRUPT) Rxisr()
        CALL    Rxisr
SkipRx:
        LAC     ONE,TXINT_BIT
        AND     IFimage
        BZ      SkipTx              ;IF (TRANSMIT PORT INTERRUPT) Txisr()
        CALL    Txisr
SkipTx:
        LAC     ONE,WDINT_BIT
        AND     IFimage
        BZ      SkipWDT             ;IF (WDT INTERRUPT) WDTisr
        CALL    WDTisr
SkipWDT:
        LAC     ONE,IOPINT_BIT
```

```
        AND     IFimage
        BZ      SkipIOP          ;IF (I/O PORT INTERRUPT) IOPisr
        CALL    IOPisr
SkipIOP:
        LAC     ONE,INT_BIT
        AND     IFimage
        BZ      SkipINT          ;IF (INT) INTisr
        CALL    INTisr
SkipINT:
*
*           EXIT INTERRUPT SERVICE ROUTINE
*
ISRexit:
*
*           RESTORE ENVIRONMENT
*
RESTORE:
        OUT     BankSave,BSR     ;RESTORE BSR
        ZALH    ACCH             ;RESTORE UPPER ACCUMULATOR
        ADDS    ACCL             ;RESTORE LOWER ACCUMULATOR
        LST     STATUS           ;RESTORE STATUS
        EINT                     ;ENABLE INTERRUPTS
        RET                      ;RETURN TO INTERRUPTED CODE
*
        END
```

5.2.3 TMS320C17 Interrupt Service Routines

The TMS320C17 has four maskable interrupts: $\overline{\text{EXINT}}$, $\overline{\text{FSR}}$, $\overline{\text{FSX}}$, and FR. The interrupts are maskable via the system control register bits CR7–CR4. Bits CR3–CR0 serve as the interrupt flags for the four interrupts. An active signal on any of these interrupts sets the corresponding interrupt flag to one. Because all four interrupts activate a single master interrupt flag, the interrupt service routine (ISR) should poll all four interrupt flags and check for the corresponding interrupt source. The ISR may also need to poll the individual mask bits (CR7–CR4) before recognizing the interrupt flag.

Interrupt processing on the TMS320C17 begins as follows:

1) The EINT (enable interrupt) instruction is executed, which sets the INTM (interrupt mode) bit to 0 so that an interrupt can be received.
2) When an interrupt occurs, the INTF (interrupt flag) bit is set to 1.

As interrupt servicing begins, the following sequence occurs automatically:

1) The interrupt is acknowledged, which clears the INTF (interrupt flag) bit to 0.
2) The INTM (interrupt mode) bit is set to 1 to disable further interrupts.
3) The current PC is pushed onto the TOS (top of stack).
4) The new PC is set to 2.

During servicing of the interrupt, perform the following operations in software:

1) Execute the interrupt service routine. Program memory address has either a service routine to save the context of the machine or a branch to the interrupt service routine.The context of the machine may be saved and restored later if required. Follow these steps to select which interrupt to service:

   a) Use software polling techniques to determine which one of the four flags has been set in the control register.

   b) Check for corresponding mask bits before proceeding (optional).

   c) Clear that flag (reset by writing a 1) and service the source of that flag. Before you clear the flag there must be an interval of at least four clock cycles after the flag has been set. Before clearing the $\overline{\text{EXINT}}$ flag, the interrupt source must have been taken away for four cycles.

All interrupts are synchronized and multiplexed into the master interrupt circuitry and have the same priority. However, you can set priorities in polling the interrupt flags. The ISR should clear the interrupt flag before executing an EINT instruction or enabling the interrupts. Note that writing a one to an interrupt flag will clear it: that is, set the corresponding flag to zero. If the interrupt condition persists when an attempt is made to clear the flag, that interrupt flag will remain set. This condition is applicable only to $\overline{\text{EXINT}}$ or its equivalent in coprocessor port mode. In the coprocessor mode on the TMS320C17, the $\overline{\text{BIO}}$ and $\overline{\text{EXINT}}$ lines cannot be driven externally but are reserved for transfers to/from the coprocessor port. Example 5–7 shows an example interrupt service routine, including polling, for a system with three active interrupts enabled

Example 5–7. TMS320C17 Interrupt Service Routine

```
* THIS ROUTINE MAY BE LOCATED AT PROGRAM MEMORY LOCATION 2,
* OR A BRANCH INSTRUCTION AT LOCATION 2 DIRECTS PROGRAM
* EXECUTION HERE. MEMORY LOCATION ONE CONTAINS THE
* CONSTANT 1. STATUS IS ALWAYS STORED ON DATA PAGE 1 WHEN
* DIRECT MEMORY ADDRESSING IS USED.
*
* RECV IS THE SERVICE ROUTINE FOR THE RECEIVE INTERRUPT.
* XINT IS THE SERVICE ROUTINE FOR THE EXTERNAL INTERRUPT.
* TRANS IS THE SERVICE ROUTINE FOR THE TRANSMIT INTERRUPT.
*
       .def    ISR,RECV,XINT
       .ref    TRANS
*
STATUS .set    0              ;
ACCL   .set    1              ;ASSIGN LOCATION TO SAVE STATUS/ACC
ACCH   .set    2              ;
RBUF   .set    3              ;STORE RECEIVE DATA HERE
CREG   .set    4              ;TEMP LOCATION TO STORE CONTROL REG
*
       .text
ISR    SST     STATUS         ;SAVE STATUS
       LDPK    1              ;USE DATA PAGE 1
SACL   ACCL                   ;SAVE LOW ACCUMULATOR
SACH   ACCH                   ;SAVE HIGH ACCUMULATOR
*
* THIS ROUTINE CHECKS FOR THREE ACTIVE INTERRUPTS OCCURRING
* AND SERVICES THEM ACCORDINGLY. IT IS ASSUMED THAT ONE OF
* THREE IS THE SOURCE OF THE INTERRUPT. AFTER AN INTERRUPT
* FLAG IS SET, IT MUST BE RESET BY THE INTERRUPT SERVICE
* ROUTINE TO AVOID BEING INTERRUPTED AGAIN ON THE RETURN
* FROM THE SUBROUTINE.
*
       IN      CREG,PA0       ;READ LOWER CONTROL REGISTER
       LAC     ONE,0          ;LOAD INT INTERRUPT MASK
       AND     CREG           ;INT FLAG SET?
       BNZ     XINT           ;GO TO INT SERVICE ROUTINE
       LAC     ONE,2          ;LOAD FSX INTERRUPT MASK
       AND     CREG           ;FSX FLAG SET?
       BNZ     TRANS          ;GO TO TRANSMIT SERVICE ROUTINE
*
```

```
* INTERRUPT MUST BE FSR.
*
RECV    LACK   0Fh           ;SET ALL INTERRUPT FLAGS IN CREG
        OR     CREG
        SACL   CREG
        LACK   0Bh           ;ZERO ALL INTERRUPT FLAGS EXCEPT FSR
        XOR    CREG
        SACL   CREG
        IN     RBUF,PA1      ;READ REC DATA FROM PORT 1
*
* RESTORE STATUS.
*
RESTOR OUT     CREG,PA0      ;RESTORE CNTL REG. CLEAR INTERRUPTS
        ZALH   ACCH          ;RESTORE HIGH ACCUMULATOR
        ADDS   ACCL          ;RESTORE LOW ACCUMULATOR
        LST    STATUS        ;RESTORE STATUS
        EINT                 ;ENABLE INTERRUPTS
        RET
*
* INTERRUPT MUST BE COPROCESSOR EXINT.
*
XINT    IN     CPBUF,PA5     ;READ LATCH DATA FROM PORT 5
        LACK   0Fh           ;SET ALL INTERRUPT FLAGS IN CREG
        OR     CREG
        SACL   CREG
        LACK   0Eh           ;ZERO ALL INTERRUPT  FLAGS EXCEPT EXINT
        XOR    CREG
        SACL   CREG
        B      RESTOR        ;BRANCH TO RESTORE STATUS
```

5.2.4 $\overline{\text{BIO}}$ Polling

A low priority interrupt can be serviced by using $\overline{\text{BIO}}$ polling. Executing the BIOZ instruction polls (or tests) the $\overline{\text{BIO}}$ pin to see if a device needs to be serviced. This method allows a critical loop or set of instructions to be executed without a variation in execution time. Because the test for the $\overline{\text{BIO}}$ pin occurs at defined points in the program, context saves are minimal.

The $\overline{\text{BIO}}$ pin can be used to monitor the status of a peripheral. If the FIFO (first in, first out) full status line is connected to the $\overline{\text{BIO}}$ pin, the FIFO is serviced only when the FIFO is full. In the following code segment, the FIFO contains 16 data words. The $\overline{\text{BIO}}$ pin is tested after each time-critical function has been executed.

```
        BIOZ SKIP
        CALL SERVE
SKIP    .
        .
        .
```

The subroutine does not have to save the registers or the status, because a new procedure will be executed after the device is serviced, as shown in the following code:

```
SERVE   LARK AR0,15
        LARK AR1,TABLE
LOOP    LARP 1
        IN   *+,PA0,AR0
        BANZ LOOP
        RET
```

The FIFO must be serviced before another word is input; otherwise, data may be lost. This servicing of the FIFO determines the frequency at which the polling must take place.

5.2.5 Context Switching

Context switching, commonly required when processing a subroutine call or interrupt, may be quite extensive or simple, depending on system requirements such as the use of the stack or auxiliary registers. Unless the interrupt service routine (ISR) is a simple I/O handler, the processing in the ISR generally must assure that the processor context is preserved during execution. The context must be saved before executing the routine itself and must be restored when the routine is finished. A common routine may be used to secure the context of the processor during interrupt processing.

The TMS320C1x program counter is stored automatically on the hardware stack. If there is any important information in the other TMS320C1x registers, such as the status or auxiliary registers, these must be saved by user software. A stack in data memory, identified by an auxiliary register, is useful for storing the machine state when processing interrupts.

During an interrupt, all registers except the P register can be saved and restored directly. However, the TMS320C1x devices have hardware protection against servicing an interrupt between an MPY or MPYK instruction and the following instruction. For this reason, it is advisable to follow the MPY and MPYK instructions with LTA, LTD, PAC, APAC, or SPAC instructions that transfer data from the P register to the accumulator.

Example 5–8 and Example 5–9 show the routine for saving and restoring the state of the TMS320C1x processor. Auxiliary register 1 (AR1) is used in both examples as the stack pointer. As the stack grows, it expands into lower memory addresses. The registers saved are the ST status register, accumulator (ACC), P register, T register, all four levels of the hardware stack, and auxiliary registers AR0 and AR1.

The routines in Example 5–8 and Example 5–9 are protected against interrupts, allowing context switches to be nested. This is done with the MAR*– and MAR*+ instructions at the beginning of the context save and context restore routines, respectively. Note that the last instruction of the context save decreases AR1 while the context restore is completed with an increase of AR1. This prevents the loss of data if a context save or restore routine is interrupted.

Example 5–8. Context Save

```
        .title    'CONTEXT SAVE'
        .def      SAVE
        .text
*
* CONTEXT SAVE ON SUBROUTINE CALL OR INTERRUPT. ASSUME THAT
* AR1 IS THE STACK POINTER AND AR1 = 128.
*
SAVE    LARP AR1        ;CHANGE POINTER TO AR1 AR1 = 128
        MAR  *-         ;                      AR1 = 127
*
* SAVE THE STATUS REGISTER.
*
        SST  *-         ;ST --> (127),         AR1 = 126
*
* SAVE THE ACCUMULATOR.
*
        SACH *-         ;ACCH --> (126),       AR1 = 125
        SACL *-         ;ACCL --> (125),       AR1 = 124
*
* SAVE THE P REGISTER.
*
* THE P REGISTER CANNOT BE EASILY RESTORED FROM MEMORY. ON
* TMS320C1x DEVICES, IT IS ASSUMED THAT THE MPY AND MPYK
* INSTRUCTIONS HAVE BEEN FOLLOWED BY AN APAC, PAC, SPAC,
* LTA, OR LTD INSTRUCTION. HENCE, SAVING THE ACCUMULATOR
* HAS ALSO SAVED THE P REGISTER.
*
* SAVE THE T REGISTER.
*
        MPYK 1          ;T --> P
        PAC             ;T --> ACC
        SACL *-         ;T --> (124),          AR1 = 123
*
* SAVE ALL FOUR LEVELS OF THE HARDWARE STACK.
*
        POP             ;TOS      --> ACC,
        SACL *-         ;TOS (4) --> (123),    AR1 = 122
        POP             ;STACK(3) --> ACC,
        SACL *-         ;STACK(3) --> (122),   AR1 = 121
        POP             ;STACK(2) --> ACC,
        SACL *-         ;STACK(2) --> (121),   AR1 = 120
        POP             ;BOS(1)   --> ACC,
        SACL *-         ;BOS(1)   --> (120),   AR1 = 119
*
* SAVE AUXILIARY REGISTERS
*
        SAR  AR0,*-     ;AR0 --> (119),    AR1 = 118
        SAR  AR1,*-     ;AR1 --> (118),    AR1 = 117
*
* SAVE IS COMPLETE.
```

Example 5–9. Context Restore

```
        .title 'CONTEXT RESTORE'
        .def   RESTOR
        .text
*
* CONTEXT RESTORE AT THE END OF A SUBROUTINE OR INTERRUPT.
* ASSUME THAT AR1 IS THE STACK POINTER AND AR1 = 117.
*
RESTOR LARP AR1         ;CHANGE POINTER TO AR1, AR1 = 117
       MAR  *+          ;                       AR1 = 118
*
* RESTORE AUXILIARY REGISTERS.
*
       LAR  AR1,*+      ;(118) --> AR1,         AR1 = 119
       LAR  AR0,*+      ;(119) --> AR0,         AR0 = 120
*
* RESTORE ALL FOUR LEVELS OF THE HARDWARE STACK.
*
       ZALS *+          ;(120) --> ACC,         AR1 = 121
       PUSH             ;(120) --> BOS (1),
       ZALS *+          ;(121) --> ACC,         AR1 = 122
       PUSH             ;(121) --> STACK(2),
       ZALS *+          ;(122) --> ACC,         AR1 = 123
       PUSH             ;(122) --> STACK(3),
       ZALS *+          ;(123) --> ACC,         AR1 = 124
       PUSH             ;(123) --> TOS (4),
*
* RESTORE THE T REGISTER.
*
       LT   *+          ;(124) --> T,           AR1 = 125
*
* RESTORE THE ACCUMULATOR.
*
       ZALS *+          ;(125) --> ACCL,        AR1 = 126
       ADDH *+          ;(126) --> ACCH,        AR1 = 127
*
* RESTORE THE STATUS REGISTER.
*
       LST  *+          ;(127) --> ST,          AR1 = 128
*
* RESTORE IS COMPLETE.
*
       EINT             ;ENABLE INTERRUPTS
       RET              ;RETURN TO CALLING ROUTINE
```

5.3 Program Control

To facilitate the use of the TMS320C1x in general-purpose high-speed processing, a variety of instructions are provided for software stack expansion, implementation of subroutine calls, addressing and loop control with auxiliary registers, and computed GOTOs. Descriptions and examples of how to use these features are given in this section.

5.3.1 Software Stack Expansion

The TMS320C1x except the TMS320C16 has a 12-bit program counter (PC) and a four-level hardware stack for PC storage. Provisions have been made on the TMS320C1x for extending the hardware stack into data memory. This is useful for deep subroutine nesting or stack overflow protection.

Note:

Because of the large (64K) address space of the 'C16, the PC is 16 bits long and the hardware stack is eight levels deep.

You can access the hardware stack via the accumulator with the PUSH and POP instructions. The PUSH instruction pushes the 12 LSBs of the accumulator onto the top of the stack (TOS). The POP instruction pops the TOS into the 12 LSBs of the accumulator. Following the POP instruction, the TOS can be moved into data memory by storing the low-order accumulator word (SACL instruction). This allows expansion of the stack into the data RAM. From data RAM, the stack can easily be copied into off-chip program RAM by using the TBLW instruction. In this way, the stack can be expanded to a very large size.

When the stack has four values stored on it and one or more values are to be put on the stack before any other values are popped off, a subroutine can be used to perform software stack expansion. Such a routine is illustrated in Example 5–10. In this example, the main program stores the stack starting location in memory in the auxiliary register and indicates to the subroutine whether to push data from memory onto the stack or pop data from the stack to memory. If a zero is loaded into the accumulator before the subroutine is called, the subroutine pushes data from memory to the stack. If a one is loaded into the accumulator, the subroutine pops data from the stack to memory.

A CALL instruction should be used to initiate execution of the software stack expansion routine. Because the CALL instruction uses the stack to save the program counter, the subroutine pops this value into the accumulator and saves it in a memory location. Then, at the end of the subroutine, this value is reloaded into the accumulator, and the main program is reentered by using the RET instruction. This prevents the calling routine program counter from being stored into a memory location. The BANZ (branch on auxiliary register not zero) instruction controls all of the loops in Example 5–10.

Example 5–10. Software Stack Expansion

```
* THIS ROUTINE EXPANDS THE STACK WHILE LETTING THE MAIN
* PROGRAM DETERMINE WHERE TO STORE THE STACK CONTENTS OR
* FROM WHERE TO RECOVER THEM.
*
LOC1    .set 0
*
STACK   LARK AR1,3      ;LOAD COUNTER
        LDPK 1          ;USE PAGE 1
        BNZ  PO         ;IF POPD IS NEEDED, GO TO PO
        POP             ;LOAD PC INTO ACCUMULATOR
        SACL LOC1       ;STORE PC AT MEM LOCATION LOC1
*
P       LARP 0          ;USE AR0
        LAC  *+,AR1     ;LOAD ACCUMULATOR INTO MEMORY
        PUSH            ;PUT MEMORY ON STACK
        BANZ P          ;BRANCH TO P UNTIL STACK IS FULL
        LAC  LOC1       ;LOAD PC INTO ACCUMULATOR
        PUSH            ;PUT RETURN ADDRESS ON STACK
        RET             ;RETURN TO MAIN PROGRAM
*
PO      POP             ;LOAD PC INTO ACCUMULATOR
        SACL LOC1       ;SAVE PC INTO MEMORY
        LARP AR0        ;USE AR0
        MAR  *-         ;ALIGN STACK POINTER
*
PO1     LARP 0          ;USE AR0
        POP             ;PUT STACK IN ACCUMULATOR
        SACL *-,0,AR1   ;STORE STACK IN MEMORY
        BANZ PO1        ;BRANCH TO PO1 UNTIL SAVED
        MAR  *+         ;REALIGN STACK POINTER
        LAC  LOC1       ;LOAD ACCUMULATOR WITH PC
        PUSH            ;PUT RETURN ADDRESS ON STACK
        RET             ;RETURN TO MAIN PROGRAM
```

5.3.2 Subroutine Calls

When a subroutine call is made with the CALL or CALA instruction, the current contents of the program counter are stored on the top of the stack. At the end of the subroutine, a RET (return from subroutine) instruction pops the top of the stack to the program counter. The program then resumes when the instruction following the subroutine call executes.

In two circumstances, a level of stack must be reserved for the machine's use. First, the TBLR and TBLW instructions use one level of stack. Second, when interrupts are enabled, the PC is saved on the stack during the interrupt routine. If a system is designed to use both interrupts and a TBLR or TBLW instruction, only two levels of stack are available for nesting subroutine calls.

Subroutine calls can be nested more than two levels deep if the return address is removed from the stack and saved in data memory. The POP instruction moves the top of stack (TOS) into the accumulator and pops the stack up one level. The return address can then be stored in data memory until the end of the subroutine when it is put back into the accumulator. The PUSH instruction pushes the stack down one level and then moves the accumulator onto the TOS. Therefore, when the RET instruction is executed, the PC is updated with the return address. This procedure allows a second subroutine to be called inside the first subroutine without using another level of stack.

The POP and PUSH instructions can also be used to pass arguments to a subroutine. The DATA directives following the subroutine call can create a list of constants and/or variables to be passed to the subroutine. After the subroutine is called, the TOS points to the list of arguments following the CALL instruction. If the argument pointer is moved from the TOS to the accumulator, the list of arguments can be read into data memory by using the TBLR instruction. Between each TBLR instruction, the accumulator must be incremented by one to point to the next argument in the list. To create the return address, the argument pointer is incremented past the last element in the argument list. The PUSH instruction moves the return address onto the TOS, and the RET instruction updates the PC. Example 5–11 illustrates a call that passes two arguments to a subroutine.

Example 5–11. Two Arguments Passed to a Subroutine

```
CLEAR BITS
*
* THIS ROUTINE CLEARS THE BITS OF A DATA WORD DESIGNATED BY
* A MASK. THE BITS SET TO ONE IN THE MASK INDICATE THE BITS
* IN THE DATA WORD TO BE CLEARED. ALL OTHER BITS REMAIN
* UNCHANGED. LOCATION ONE CONTAINS THE CONSTANT 1. MINUS
* CONTAINS A MASK INVERTER, -1 OR 0FFFFh. TWO ARGUMENTS ARE
* PASSED TO THIS SUBROUTINE. THE CALLING SEQUENCE IS AS
* FOLLOWS:
*
* CALL CBITS
       .data  VALUE     ;1ST AURGUMENT = ADDRESS OF DATA WORD
       .data  0081h     ;2ND ARGUMENT = MASK
*
STATUS .set   0         ;STORE STATUS REGISTER HERE
XR0    .set   126       ;TEMPORARY LOCATIONS
XR1    .set   127
*
CBITS  SST    STATUS    ;SAVE STATUS
       LDPK   0         ;USE DATA PAGE 0
       SAR    AR0,XR0   ;SAVE AR0 IN TEMPORARY LOCATION
*
       POP              ;GET ADDRESS OF 1ST ARG. IN ACC
       TBLR   XR1       ;STORE 1ST ARG. IN TEMP LOCATION
       LAR    AR0,XR1   ;PUT 1ST ARGUMENT INTO AR0
       ADD    ONE       ;POINT TO 2ND ARGUMENT
       TBLR   XR1       ;2ND ARGUMENT = MASK
       ADD    ONE       ;POINT TO RETURN ADDRESS
       PUSH             ;PUT RETURN ADDRESS ON TOS
*
       LARP   0
       LAC    XR1       ;LOAD MASK INTO ACCUMULATOR
       XOR    MINUS     ;INVERT MASK
       AND    *         ;CLEAR BITS
       SACL   *         ;STORE MODIFIED VALUE
*
       LAR    AR0,XR0   ;RESTORE AR0
       LDPK   1         ;USE DATA PAGE 1
       LST    STATUS    ;RESTORE STATUS REGISTER
       RET              ;RETURN TO MAIN PROGRAM
```

Hardware stack allocation involves allocating the usage of the various stack levels for interrupts, subroutine calls, pipelined instructions, and the emulator (XDS). The TMS320C1x disables all interrupts when it takes an interrupt trap. If interrupts are enabled more than one instruction before the return of the interrupt service routine, the routine can also be interrupted; this uses another level of the hardware stack. Take this into consideration when managing the use of the stack.

During nesting of subroutine calls, each call uses a level of the stack. The number of levels used by interrupts, as well as the depth of the nesting of subroutines must be considered. Here are two possible allocations of the hardware stack:

- 1 level reserved for interrupt service routines (ISR)
- 3 levels available for subroutine calls

or:

- 1 level reserved for interrupt service routines (ISR)
- 2 levels available for subroutine calls
- 1 level available for TBLR/TBLW instructions

5.3.3 Addressing and Loop Control With Auxiliary Registers

The two auxiliary registers on the TMS320C1x can be used either as pointers for indirect addressing or as loop counters. In the indirect addressing mode, the auxiliary register pointer (ARP) determines which auxiliary register is selected. The LARP instruction sets the ARP equal to the value of the immediate operand. The value of the ARP can also be changed in the indirect addressing mode; the ARP is updated after the instruction has been executed.

The contents of the auxiliary register are interpreted as a data memory address when the indirect addressing mode is used. A sequential list of data can easily be accessed in the indirect mode by using the autoincrement/decrement feature of the auxiliary registers. The auxiliary register can also be used as a 9-bit counter (see subsection 3.3.5). The MAR (modify auxiliary register and pointer) instruction allows the auxiliary register selected by the ARP to be incremented or decremented without implementing any other operation in parallel.

Three instructions (LARK, LAR, and SAR) either load or store a value into an auxiliary register, independently of the value of the ARP. The first operand in each of these instructions determines which auxiliary register is to be either loaded or stored. This operand does not affect the value of the ARP for subsequent instructions.

Example 5–12 illustrates using an auxiliary register in the indirect addressing mode to input data into a block of memory.

Example 5–12. Auxiliary Register Indirect Addressing

```
* THIS ROUTINE USES AN AUXILIARY REGISTER IN THE INDIRECT
* ADDRESSING MODE TO INPUT DATA INTO A BLOCK OF MEMORY.
*
       LARK AR0,DATBLK     ;INIT AR0 AS A POINTER TO DATBLK
*                          ;(AREA OF 8 WORDS IN DATA MEMORY)
       LARP 0              ;SELECT AR0
       LACK 8              ;INIT ACCUMULATOR AS A COUNTER
*
LOOP   IN   *+,PA0         ;INPUT DATA
       SUB  ONE            ;DECREASE COUNTER (ONE = VALUE 1)
       BNZ  LOOP           ;REPEAT UNTIL COUNT = 0
```

When an auxiliary register is used as a loop counter, the BANZ instruction tests and then decreases the auxiliary register selected by ARP. Because the test for zero occurs before the auxiliary register is decreased, the value loaded into the auxiliary register must be one less than the number of times the loop should be executed. The maximum number of loops that can be counted is 512 because only 9 bits of each auxiliary register are implemented as counters. A routine that inputs data and calculates a sum while the auxiliary register is used to count the number of loops is shown in Example 5–13. The accumulator contains the result.

Example 5–13. Auxiliary Register Loop Counting

```
* THIS ROUTINE USES AN AUXILIARY REGISTER TO COUNT THE
* NUMBER OF LOOPS.
*
       LARK AR0,3          ;INITIALIZE AR0 AS A COUNTER
       LARP 0              ;SELECT AR0
       ZAC                 ;CLEAR ACCUMULATOR
*
LOOP   IN   DATA1,PA2      ;INPUT DATA VALUE
       ADD  DATA1          ;ADD DATA TO ACCUMULATOR
       BANZ LOOP           ;REPEAT LOOP FOUR TIMES
```

Both indirect addressing and loop counting can be performed at the same time to implement loops efficiently. If the data block is defined to start at location 0 in data memory, the same auxiliary register that is counting the number of loops can also be the pointer for indirect addressing, as shown below. Note that data locations 0 through 7 are loaded with input data.

```
       LARK AR0,7          ;AR0 POINTS TO END OF DATA BLOCK
LOOP   IN   *,PA2          ;INPUT DATA VALUE
       BANZ LOOP           ;REPEAT LOOP 8 TIMES
```

The data block does not have to start at zero if one auxiliary register is used for counting and the other register is used as a pointer. Example 5–14 illustrates how both auxiliary registers can be used at once.

Example 5–14. Auxiliary Register Pointing and Loop Counting

```
* THIS ROUTINE USES ONE AUXILIARY REGISTER FOR POINTING AND
* THE OTHER REGISTER FOR LOOP COUNTING.
*
       LARK AR0,7          ;INITIALIZE AR0 AS A COUNTER
       LARK AR1,DATBLK     ;AR0 POINTS TO START OF DATBLK
*                          ; (DATA MEMORY AREA)
       ZAC                 ;CLEAR ACCUMULATOR
*
LOOP   LARP 1              ;POINT TO AR1
       ADD  *+,AR0         ;CALCULATE SUM OF DATA IN BLOCK
*                          ;POINT TO AR0
       BANZ LOOP           ;REPEAT LOOP 8 TIMES
```

5.3.4 Computed GOTOs

Processing may be executed in a time-dependent (interrupt-driven) or process-dependent (user-selected) way. Selecting the processing mode may depend on the result of a particular computation. A simple computed GOTO can be programmed in the TMS320C1x by using the CALA instruction. This instruction uses the contents of the accumulator as the direct address of the call. The address of the subroutine can be computed from a data value to determine which one of several routines is executed. The return at the end of each of these routines causes program execution to resume with the instruction following the CALA command. Note that the CALA instruction uses a level of stack because it is an indirect subroutine call, not just an indirect branch.

Example 5–15 illustrates how to compute a call to one of several routines. The subroutines are defined first, and then a table of branches to each subroutine is created. The main part of the program inputs a data value of 0, 1, or 2. The appropriate address in the table is calculated in the accumulator. An indirect subroutine call causes the proper branch in the table to be executed.

Example 5–15. Computed GOTO

```
* THIS ROUTINE COMPUTES AND EXECUTES A SUBROUTINE CALL.
ONE     .set  126           ;STORE CONSTANT 1
VALUE   .set  127           ;VALUE READ FROM PORT 4
SUB1    IN    DAT1,PA0      ;INPUT DATA VALUE FROM PORT 0
        RET
*
SUB2    IN    DAT1,PA1      ;INPUT DATA VALUE FROM PORT 1
        RET
*
SUB3    IN    DAT1,PA2      ;INPUT DATA VALUE FROM PORT 2
        RET
*
TBL1    B     SUB1          ;CREATE TABLE OF BRANCHES TO EACH
        B     SUB2          ;SUBROUTINE DEFINED
        B     SUB3
*
START   LDPK  0             ;USE PAGE 0
        LACK  1             ;ACC = 1
        SACL  ONE           ;STORE 1 IN LOCATION ONE
        LTONE               ;LOAD T REGISTER WITH VALUE OF 1
        MPYK  TBL1          ;GET ADDRESS OF TABLE
        PAC
        IN    VALUE,PA4     ;INPUT DATA VALUE OF 0, 1, OR 2
        LT    VALUE         ;LOAD T REG WITH VALUE FROM PA4
        MPYK  2             ;CALCULATE OFFSET
        APAC
        CALA                ;GO TO DESIGNATED SUBROUTINE
        LAC   DAT1          ;RETURN HERE AFTER SUBROUTINE
         .
         .
         .
```

5.4 Memory Management

In the TMS320C1x's modified Harvard architecture, with program and data memories in two separate spaces, the next instruction fetch can occur while the current instruction is fetching data and executing the operation. This increases the speed of the device, but it requires the use of instructions to transfer a word between data memory and program memory.

Data memory consists of 144/256 words of 16-bit on-chip RAM with all nonimmediate data operands residing within this RAM. Program memory consists of 1.5K/4K words of 16-bit on-chip ROM with 1524/4000 words reserved for program use. Only those devices with EPROM capability can access all 4096 words. Because there is no microprocessor mode of operation on the TMS320C17, all program memory resides within the on-chip ROM.

The TMS320C1x uses three forms of data memory addressing: direct, indirect, and immediate. Direct addressing uses the seven lower bits of the instruction word concatenated with the data page pointer to form the data memory address. Indirect addressing uses the lower eight bits of the auxiliary registers as the data memory address. Immediate addressing uses part of the instruction word for data addressing rather than using RAM.

The structure of the TMS320C1x memory map can vary for each application (see subsection 3.3.4 for memory maps). Instructions can move data and constants into data memory. Explanations and examples are provided in this section.

The TMS320C14 provides two options of program memory usage: the 4K words of internal ROM/EPROM can be used in the microcomputer mode or in the microprocessor mode. These modes can be configured in hardware or in software. The hardware method uses the $\overline{NMI}$/MC/$\overline{MP}$ and $\overline{RS}$ pins to set the mode. The software method uses bit 0 of the SYSCON register to set the mode. A one written to this bit sets the CPU in the microprocessor mode.

For the TMS320C14, the initial configuration of the CPU is not binding; that is, it can be changed later into the other mode and then changed back if desired. This effectively doubles the amount of program memory to 8K words. In Example 5–16, the CPU is set to the microprocessor mode.

Example 5–16. TMS320C14 Memory Expansion Routine

```
*
* THIS IS AN EXAMPLE OF HOW TO MANIPULATE THE MC/MP BIT IN THE
* SYSCON REGISTER TO FORCE MICROPROCESSOR MODE: THAT IS, TO
* FORCE PROGRAM ACCESS TO EXTERNAL MEMORY.
*
      .include"C14INC"    ;INCLUDE HEADER FILE
*
* IT IS ASSUMED THAT BANK AND TMP ARE ALLOCATED IN THE CURRENT
* MEMORY.
*
      .ref    BANK,TMP
*
* IT IS ALSO ASSUMED THAT 'EXTERNAL' IS THE ADDRESS IN
* EXTERNAL MEMORY WHICH IS TO BE EXECUTED AFTER THE MC/MP
* BIT CHANGE.
*
      .ref    EXTERNAL
*
      .text
*
      .
      .
      .
*
      LACK    SYSBank
      SACL    BANK
      OUT     BANK,BSR
      ZAC
      SACL    TMP
      OUT     TMP,SYSCON
      CALL    EXTERNAL   ;CALL SUBROUTINE IN EXTERNAL MEMORY.
                         ;PROGRAM FLOW RETURNS HERE IF
                         ;MC/MP BIT IS SET TO 1 PRIOR
                         ;TO THE 'RET' INSTRUCTION
      .end
```

5.4.1 Moving Data

The DMOV (data move) instruction allows a data word to be written into the next higher memory location in a single cycle without affecting the accumulator. If variables are placed in consecutive locations, a DMOV instruction can be used to move each of the variables before the next calculation is performed. For example, when a digital filter is implemented, the variables in the equation represent the inputs and outputs at discrete times. This type of data structure is typically implemented as a shift register when the data at time t is shifted to the position previously occupied by the data at time t–1. If consecutive addresses in data memory correspond to consecutive time increments, then the DMOV instruction can move the data item at location d to location d+1, thereby accomplishing a shift. The DMOV feature is useful in implementing filters and convolution algorithms.

The LTD instruction combines the data move operation with the LTA (load T register and accumulate previous product) instruction operations, performing the three operations in parallel. The operand of the instruction is loaded into

the T register; the operand is also written into the next higher memory location and the P register is added to the accumulator. Using the LTD instruction makes the order of the multiply and accumulate operations important because the data is being moved while the calculation is being performed. The oldest input variable must be multiplied by its constant and loaded into the accumulator first. Then the input, which is one time-unit delay less, is multiplied and accumulated. This process is repeated until the entire equation has been computed.

Example 5–17 illustrates the use of the LTD instruction to move input variables in memory as the results are calculated.

Example 5–17. Moving Data Using the LTD Instruction

```
* THE FOLLOWING EQUATION WILL BE IMPLEMENTED TO DEMONSTRATE
* THE USE OF THE LTD INSTRUCTION. AT THE END OF THE SUB-
* ROUTINE, LOCATION X1 IS AVAILABLE TO INPUT THE NEW SAMPLE.
*
* Y = A*X3 + B*X2 + C*X1
*
* WHERE A, B, C, X1, X2, AND X3 ARE VALUES STORED AT THESE
* ADDRESSES.
*
X1      .set   0    ;USE THESE MEMORY LOCATIONS
X2      .set   1
X3      .set   2
Y       .set   3
A       .set   127
B       .set   126
C       .set   125
*
START   ZAC            ;CLEAR ACCUMULATOR
        LDPK   0       ;USE PAGE 0
        LT     X3
        MPY    A       ;P = A*X3
        LTD    X2      ;T = X2, X2 --> X3, ACC = A*X3
        MPY    B       ;P = B*X2
        LTD    X1      ;T = X1, X1 --> X2, ACC = A*X3 + B*X2
        MPY    C       ;P = C*X1
        APAC           ;ACC = A*X3 + B*X2 + C*X1
        SACH   Y,1     ;Y = ACCH
```

The table below illustrates the effect on data memory after execution of the code in Example 5–17.

Data Memory	Before Instruction	After Instruction
0h	X1	X1
1h	X2	X1
2h	X3	X2

5.4.2 Moving Constants Into Data Memory

Most signal processors have a separate memory space for storing constants. By supporting communication between data and program memory, the TMS320C1x incorporates constant memory capability with its program memory and uses memory space efficiently. The portion of memory not used for storing constants is available for use as program space.

Five immediate instructions efficiently execute operations with constants. The LARP instruction changes the auxiliary register pointer, and the LDPK instruction changes the data page pointer. The LACK, LARK, and MPYK instructions allow constants to be used in calculations. LACK and LARK both require an unsigned operand with a magnitude no greater than eight bits. The MPYK instruction allows a 13-bit signed number as an operand.

A 16-bit value can be moved from program memory to data memory with the TBLR instruction. TBLR obtains the program memory address (the source) from the accumulator and the data memory address (the destination) from the operand of the instruction. This instruction is commonly used to look up values in a table in program memory. The address of the value in the table is computed in the accumulator before the instruction is executed. TBLR then moves the value into data memory. TBLR is a three-cycle instruction and, therefore, takes longer than an immediate instruction. However, it has more flexibility because it operates on 16-bit constants.

Sometimes it is convenient to store data operands in program ROM or external memory and then read them into the on-chip RAM as they are needed. There are two ways to do this: First, the TBLR (table read) instruction can transfer data from on-chip program ROM to on-chip data RAM. Second, off-chip data RAM can be addressed via the IN and OUT instructions. The IN and OUT instructions can read and write from data RAM to large amounts of external storage addressed as a peripheral.

The TBLW instruction is another way to transfer data to program memory. The IN and OUT instructions can transfer data between the on-chip data memory and the I/O space (see Section 6.1).

Note that the TBLW (table write) instruction should not be used on the TMS320C17 because this instruction transfers data from on-chip data RAM to external memory. The TMS320C17 does not directly interface to external memory because the port address bits (PA2–PA0) are the only address lines external to the device.

Example 5–18 illustrates bringing the cosine value of a variable into data memory by using the TBLR instruction. Note that if the address of COSINE is greater than 255, the address is loaded into the accumulator by loading the T register with a one, multiplying by the constant COSINE, and transferring it from the P register into the accumulator.

Example 5–18. Using TBLR to Move a Constant Into Data Memory

```
*THIS ROUTINE USES THE TBLR INSTRUCTION TO BRING THE COSINE
*VALUE OF A VARIABLE INTO DATA MEMORY. A TABLE CONTAINING
*THE COSINE VALUES IS FIRST CREATED IN PROGRAM MEMORY.
*
COSINE DATA


START   IN    X,PA0
        LACK  COSINE      ;LOAD TABLE ADDRESS
        ADD   X           ;CALCULATE PROGRAM MEMORY ADDRESS
        TBLR  COSX        ;MOVE VALUE INTO DATA MEMORY
```

The effect on data memory after the TBLR instruction has been executed in Example 5–18 is as follows:

	Before TBLR Execution	After TBLR Execution
Program Memory		
COSINE + X	02FFh	02FFh
Data Memory		
COSX	71F2h	02FFh

When data is transferred from program memory into data memory with the TBLR instruction, a calculated, rather than predetermined, location of data in program memory may be specified for transfer. A routine using this approach is shown in Example 5–19.

Example 5–19. Moving Program Memory to Data Memory With TBLR

```
* THIS ROUTINE USES THE TBLR INSTRUCTION TO MOVE DATA VALUES
* FROM PROGRAM MEMORY INTO DATA MEMORY. THIS ROUTINE, CAN
* SPECIFIES THE PROGRAM MEMORY LOCATION IN THE ACCUMULATOR
* FROM WHICH DATA IS TO BE MOVED TO A SPECIFIC DATA MEMORY
* LOCATION. ASSUME THAT THE ACCUMULATOR CONTAINS THE
* ADDRESS IN PROGRAM MEMORY FROM WHICH TO TRANSFER THE DATA.
*
TABLE   LARP  1           ;USE AR1
        LARK  AR1,63      ;START FROM ADDRESS 63
*
LOOP    TBLR  *           ;MOVE DATA INTO DATA RAM
        BANZ  LOOP        ;TRANSFER 64 VALUES
        RET               ;RETURN TO CALLING PROGRAM
```

In cases where systems require that temporary storage be allocated in the program memory, TBLW can be used to transfer data from internal data memory to external program memory. The code in Example 5–20 demonstrates how this may be accomplished.

Example 5–20. Moving Internal Data Memory to Program Memory With TBLW

```
* THIS ROUTINE USES THE TBLW INSTRUCTION TO MOVE DATA VALUES
* FROM INTERNAL DATA MEMORY TO EXTERNAL PROGRAM MEMORY. THE
* CALLING ROUTINE MUST SPECIFY THE DESTINATION PROGRAM
* MEMORY ADDRESS IN THE ACCUMULATOR. ASSUME THAT THE
* ACCUMULATOR CONTAINS THE ADDRESS IN PROGRAM MEMORY INTO
* WHICH THE DATA IS TRANSFERRED.
*
TABLE   LARK AR1,63          ;LOAD LOOP COUNT OF 64
        LARK AR0,DAT1        ;LOAD STARTING ADDRESS
*
LOOP    LARP AR0             ;USE AR0
        TBLW *+,AR1          ;MOVE DATA TO EXT. PROGRAM RAM
        BANZ LOOP            ;DECREMENT AND CHECK IF DONE
        RET                  ;RETURN TO CALLING PROGRAM
```

The effect on data memory after the TBLW instruction has been executed in Example 5–20 is as follows:

	Before TBLR Execution	After TBLR Execution
Program Memory		
PROG1	0FF10h	1234h
Data Memory		
DAT1	1234h	1234h

The IN and OUT instructions are used to transfer data between the data memory and the I/O space, as shown in Example 5–21 and Example 5–22.

Example 5–21. Moving Data From I/O Space into Data Memory With IN

```
* THIS ROUTINE USES THE IN INSTRUCTION TO MOVE DATA VALUES
* FROM THE I/O SPACE INTO DATA MEMORY. DATA ACCESSED FROM
* I/O PORT 7 IS TRANSFERRED TO SUCCESSIVE MEMORY LOCATIONS
* ON DATA PAGE 0.
*
INPUT   LARK AR0,32          ;SET UP LOOP COUNT
        LARK AR1,DAT1        ;SET UP DESTINATION ADDRESS
*
LOOP    LARP AR1             ;USE AR1
        IN   *+,PA7,AR0      ;MOVE DATA INTO DATA RAM
        BANZ LOOP            ;DECREMENT AND CHECK IF DONE
        RET                  ;RETURN TO CALLING PROGRAM
```

Example 5–22. Moving Data From Data Memory to I/O Space With OUT

```
* THIS ROUTINE USES THE OUT INSTRUCTION TO MOVE DATA VALUES
* FROM THE DATA MEMORY TO THE I/O SPACE. DATA IS TRANSFERRED
* TO I/O PORT 7 FROM SUCCESSIVE MEMORY LOCATIONS ON DATA
* PAGE 0.
*
OUTPUT LARK AR0,32           ;SET UP LOOP COUNT
       LARK AR1,DAT1         ;SET UP STARTING ADDRESS
*
LOOP   LARP AR1              ;USE AR1
       OUT  *+,PA7,AR0       ;MOVE DATA INTO I/O SPACE
       BANZ LOOP             ;DECREMENT AND CHECK IF DONE
       RET                   ;RETURN TO CALLING PROGRAM
```

5.5 Logical and Arithmetic Operations

Although the TMS320C1x instruction set is oriented toward digital signal processing, it can also perform the same fundamental operations of a general-purpose processor, such as bit manipulation, logical and arithmetic operations, logical and arithmetic shifts, and overflow management. Explanations and examples of how to use instructions for scaling, convolution operations, fixed-point multiplication/division/addition, and floating-point arithmetic are also given in this section.

The contents of the accumulator may be stored in data memory with the SACH and SACL instructions or stored in the stack with the PUSH instruction. The accumulator may be loaded from data memory with the ZALH, ZALS, and LAC instructions, which zero the accumulator before loading the data value. The ZAC instruction also zeros the accumulator. POP can restore the accumulator contents from the stack. The accumulator is also affected by the execution of the ABS instruction, which replaces the contents of the accumulator with its absolute value.

5.5.1 Bit Manipulation

A specified bit of a word from data memory can be set, cleared, or tested. Such bit manipulations are accomplished by using the hardware shifter and the logic instructions, AND, OR, and XOR. In Example 5–23, operations on single bits are performed on the data word VALUE. In this and the following example, data memory location ONE contains the value 1, and MINUS contains the value –1 (all bits set).

Example 5–23. Single-Bit Manipulation

```
* CLEAR BIT 5 OF DATA MEMORY LOCATION VALUE. MEMORY LOCATION
* ONE CONTAINS CONSTANT 1. MEMORY LOCATION MINUS CONTAINS -1
* OR 0FFFFh.
*
      LAC    ONE,5      ;ACC = 00000020h
      XOR    MINUS      ;INVERT ACCUMULATOR, ACC = 0000FFDFh
      AND    VALUE      ;BIT 5 OF VALUE IS ZEROED
      SACL   VALUE
*
* SET BIT 12 OF VALUE.
*
      LAC    ONE,12     ;ACC = 00001000h
      OR     VALUE      ;BIT 12 OF VALUE
      SACL   VALUE
*
* TEST BIT 3 OF VALUE.
*
      LAC    ONE,3      ;ACC = 00000008h
      AND    VALUE      ;TEST BIT 3 OF VALUE
      BZ     BIT3Z      ;BRANCH TO BIT3Z IF BIT IS CLEAR
```

More than one bit can be set, cleared, or tested at one time if the necessary mask exists in data memory. In Example 5–24, the six low-order bits in the word VALUE are cleared if MASK contains the value 63.

Example 5–24. Multiple-Bit Manipulation

```
*
* CLEAR LOWER SIX BITS OF VALUE. MEMORY LOCATION MASK
* CONTAINS THE MASK TO CLEAR THE BITS. MEMORY LOCATION
* MINUS CONTAINS -1 OR 0FFFFh.
*
      LAC    MASK   ;ACC = 0000003Fh
      XOR    MINUS  ;INVERT ACCUMULATOR; ACC = 0000FFC0h
      AND    VALUE  ;CLEAR LOWER SIX BITS
      SACL   VALUE
```

5.5.2 Overflow Management

Two TMS320C1x features can handle overflow management: the branch on overflow conditions and accumulator saturation (overflow mode). These features provide several options for overflow protection within an algorithm.

The BV (branch on overflow) instruction enables a program to branch to an error handler routine on an overflow of the accumulator. This instruction can be performed after any ALU operation that may cause an accumulator overflow.

The overflow mode is a useful feature for DSP applications. This mode simulates the saturation effect characteristic of analog systems. When enabled, any overflow in the accumulator results in the replacement of the accumulator contents by the largest positive value (7FFFFFFFh) if the overflowed number is positive, or the largest negative value (80000000h) if negative. The overflow mode is controlled by the OVM bit of the status register and can be changed by the SOVM (set overflow mode), ROVM (reset overflow mode), or LST (load status register) instructions. Overflows can be detected in software by testing the OV (overflow) bit in the status register. When a branch is used to test the overflow bit, OV is automatically reset. Note that the OV bit does not function as a carry bit. It is set only when the absolute value of a number is too large to be represented in the accumulator, and it is not reset except by specific instructions. The overflow mode feature affects all arithmetic operations in the ALU.

In Example 5–25, the accumulator saturates to 7FFFFFFFh or the largest positive value. The BV instruction also clears the OV bit.

Example 5–25. Overflow Management

```
* THE ACCUMULATOR SATURATES TO THE HIGHEST POSITIVE VALUE
* WHEN OVERFLOW OCCURS. THE ACCUMULATOR CONTAINS 7FFFF423h.
* MEMORY LOCATION A CONTAINS 74EDh. MEMORY LOCATION B
* CONTAINS 67AFh.
*
      SOVM          ;SET OVERFLOW MODE
      LT    A       ;T = 74EDh
      MPY   B       ;P = 2F5B4903h
      APAC          ;ACC = 7FFFFFFFh
      BV    OVRFLW  ;CHECK OV BIT
*                   ;BRANCH TO OVERFLOW HANDLING ROUTINE
```

The effect on the accumulator before and after the instruction execution is shown as follows:

5.5.3 Scaling

Scaling the data coming into the accumulator or already in the accumulator helps signal processing algorithms. This is frequently necessary in adaptation or other algorithms that must compute and apply correction factors or normalize intermediate results. Scaling and normalizing are implemented on the TMS320C1x via shifts of data on the incoming path to the accumulator.

There are two types of shifts: logical and arithmetic. A logical shift is implemented by filling the empty bits to the left of the MSB with zeros, regardless of the value of the MSB. An arithmetic shift fills the empty bits to the left of the MSB with ones if the MSB is one, or with zeros if the MBS is zero. The second type of bit padding is referred to as sign extension.

Data can be left-shifted 0 to 16 bits when the accumulator is loaded, and left-shifted 0, 1, or 4 bits when the SACH instruction stores from the accumulator. These shifts load numbers into the high 16 bits of the accumulator and renormalize the result of a multiply. The incoming left shift of 0 to 16 bits is supplied in the instruction itself. Left shifts of data fetched from data memory are available for loading the accumulator (LAC), adding to the accumulator (ADD), and subtracting from the accumulator (SUB). The ZALH, ADDH, and SUBH instructions are used when data is left-shifted 16 bits. The SACH instruction, which can left-shift 0, 1, or 4 bits, is used to shift out the extra sign bits for fractional multiplication (see subsection 5.5.5).

The hardware shift, which is built into the ADD, SUB, and LAC instructions, performs an arithmetic left-shift on a 16-bit word. This feature can also be used to perform right-shifts. A right-shift of n is implemented by performing a left-shift of 16–n and saving the upper word of the accumulator. Example 5–26 shows an arithmetic right-shift of 7 on a 16-bit number in the accumulator.

Example 5–26. Arithmetic Right-Shift

```
SACL   TEMP       ;MOVE NUMBER TO MEMORY
LAC    TEMP,9     ;SHIFT LEFT (16-7)
SACH   TEMP       ;SAVE HIGH WORD IN MEMORY
LAC    TEMP       ;RETURN NUMBER BACK TO ACCUMULATOR
```

The effect on the accumulator before and after the code execution is shown as follows:

A logical right-shift of 4 on a 32-bit number stored in the accumulator is shown in Example 5–26. The 32-bit results of the shift are then stored in data memory. In this example, the accumulator initially contains the hexadecimal number, 9D84C1B2h. The variables, SHIFTH and SHIFTL, receive the high word (09D8h) and low word (4C1Bh) of the shifted results.

Example 5–27. Logical Right-Shift

```
*
* SHIFT THE LOWER WORD. MEMORY LOCATION MINUS CONTAINS -1
* OR 0FFFFh.
*
   SACH   SHIFTH     ;SHIFTH = 9D84h INITIAL VALUES
   SACL   SHIFTL     ;SHIFTL = 0C1B2h
   LAC    SHIFTL,12  ;ACC = 0FC1B2000h
   SACH   SHIFTL     ;SHIFTL = 0FC1Bh
   LAC    MINUS,12   ;ACC = 0FFFFF000h
   XOR    MINUS      ;ACC = 0FFFF0FFFh
   AND    SHIFTL     ;ACC = 00000C1Bh
*
* SHIFT THE UPPER WORD.
*
   ADD    SHIFTH,12  ;ACC = 0F9D84C1Bh
   SACL   SHIFTL     ;SHIFTL = 4C1Bh FINAL LOW VALUE
   SACH   SHIFTH     ;SHIFTH = 0F9D8h
   LAC    MINUS,12   ;ACC = 0FFFFF000h
   XOR    MINUS      ;ACC = 0FFFF0FFFh
   AND    SHIFTH     ;ACC = 000009D8h
   SACL   SHIFTH     ;SHIFTH = 09D8h FINAL HIGH VALUE
```

The accumulator is affected before and after the code execution as follows:

An arithmetic right-shift of 4 can be implemented with the same routine as shown above, except with the last four lines omitted.

5.5.4 Convolution Operations

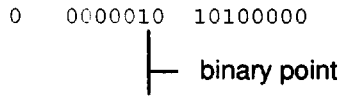
Many DSP applications must perform convolution operations or other operations similar in form. These operations require data to be shifted or delayed. The DMOV and LTD instructions can perform the data moves needed for convolution.

The data move function allows a word to be copied from the currently addressed data memory location in on-chip RAM to the next higher location while the data from the addressed location is being operated upon (for example, by the CALU). The data move and the CALU operation are performed in the same cycle. The data move function is useful in implementing algorithms, such as convolutions and digital filtering, in which data is passed through a time window. It models the z^{-1} delay operation encountered in those applications.

5.5.5 Multiplication

The TMS320C1x hardware multiplier normally performs 2s-complement 16-bit by 16-bit multiplies and produces a 32-bit result in a single processor cycle. To multiply two operands, load one operand into the T register with the multiply instruction and move the second operand to the multiplier, which then produces the product in the P register. Before another multiply can be performed, the contents of the P register must be moved to the accumulator. Pipelining the multiplies and the P-register moves allows most multiply operations to be performed with a single instruction.

Computation of nonintegers (fractional) on the TMS320C1x is based on a fixed-point 2s-complement representation of numbers. Each 16-bit number is evaluated with a sign bit, i integer bits, and 15–i fractional bits. Thus, the number



has a value of 2.625. This particular number is said to be represented in a Q8 format (8 fractional bits). Its range is between –128 (1000000000000000) and 127.996 (0111111111111111). The fractional accuracy of a Q8 number is about 0.004 (one part in 2^8 or 256).

Although particular situations (for example, a combination of dynamic range and accuracy requirements) require mixed notations, it is more common to work entirely with fractions represented in a Q15 format or integers in a Q0 format. This is especially true for signal processing algorithms where multiply and accumulate operations are dominant. The result of a fraction times a fraction remains a fraction, and the result of an integer times an integer remains an integer. No overflows are possible

Q format is a number representation commonly used when operations are performed on noninteger numbers. In Q format, the Q number (15 in Q15) denotes how many bits are located to the right of the binary point. A 16-bit number in Q15 format, therefore, has an assumed binary point immediately to the right of the most significant bit. Because the most significant bit constitutes the sign of the number, then numbers represented in Q15 may take on values from +1 (represented by +0.99997...) to –1.

A wide variety of situations may be encountered in the multiplication of two numbers. Three of these situations are provided in Example 5–28, Example 5–29 and Example 5–30.

Example 5–28. Fraction × Fraction (Q15 × Q15 = Q30)

```
                    0100000000000000  = 0.5 in Q15
                  × 0100000000000000  = 0.5 in Q15
  ------------------------------------
00 01000000000000   0000000000000000  = 0.25 in Q30
   |
   |— binary point
```

Two sign bits remain after the multiply. Generally, a single-precision (16-bit) result is saved, and the full intermediate precision is not maintained. The upper half of the result does not contain a full 15 bits of fractional precision, because the multiply operation actually creates a second sign bit. To recover that precision, the product must be shifted left by one bit, as shown in the following code excerpt:

```
LT     OP1          ;OP1 = 4000h (0.5 in Q15)
MPY    OP2          ;OP2 = 4000h (0.5 in Q15)
PAC
SACH   ANS,1        ;ANS = 2000h (0.25 in Q15)
```

The MPYK instruction provides a multiply by a 13-bit signed constant. In fractional notation, this means that a Q15 number can be multiplied by a Q12 number. The resulting number must be left-shifted by four bits to maintain full precision.

```
LT     OP1          ;OP1 = 4000h (0.5 in Q15)
MPYK   2048         ;OP2 = 0800h (0.5 in Q12)
PAC
SACH   ANS,4        ;ANS = 2000h (0.25 in Q15)
```

Example 5–29. Integer × Integer (Q0 × Q0 = Q0)

```
                    0000000000010001  = 17 in Q0
                  ×  1111111111111011  = –5 in Q0
-------------------------------------------------
1111111111111111  1111111110101011  = –85 in Q0
                                  |
                                  |– binary point
```

In this case, the extra sign bits do not change the result, and the desired product is entirely in the lower half of the product, as shown in the following program:

```
LT     OP1    ;OP1 = 0011h ( 17 in Q0)
MPY    OP2    ;OP2 = 0FFFBh ( -5  in Q0)
PAC
SACH   ANS    ;ANS = 0FFABh (-85  in Q0)
```

Example 5–30. Mixed Notation (Q14 × Q14 = Q28)

```
                    0110000000000000  =1.50 in Q14
                  ×  0011000000000000  =0.75 in Q14
-------------------------------------------------
0001  001000000000  0000000000000000  = 1.125 in Q28
    |
    |– binary point
```

The maximum magnitude of a Q14 number is just under two. Thus, the maximum magnitude of the product of two Q14 numbers is four. This possibility requires two integer bits to allow for this possibility, leaving a maximum precision for the product of 13 bits. In general, the following rule applies: The product of a number with i integer bits and f fractional bits, and a second number with j integer bits and g fractional bits is a number with (i+j) integer bits and (f+g) fractional bits. The highest precision possible for a 16-bit representation of this number has (i+j) integer bits and (15–i–j) fractional bits.

If the physical system being modeled is well understood, the precision with which the number is modeled can be increased. For example, if it is known that the above product can be no more than 1.8, the product can be represented as a Q14 number rather than as the theoretical worst case of Q13, shown in the following program:

```
LT     OP1          ;OP1 = 6000h (1.5 in Q14)
MPY    OP2          ;OP2 = 3000h (0.75 in Q14)
PAC
SACH   ANS,1        ;ANS = 2400h (1.125 in Q13)
```

The techniques illustrated in the previous three examples all truncate the result of the multiplication to the desired precision. The error generated as a result can be as much as minus one full LSB. This is true whether the truncated number is positive or negative. It is possible to implement a simple rounding technique to reduce this potential error by a factor of two, as shown in the code se-

quence of Example 5–31. The maximum error generated in this example is plus one-half LSB, whether ANS is positive or negative.

Example 5–31. Rounding Technique for Multiplication

```
LT     OP1
MPY    OP2     ;OP1 * OP2
PAC
ADD    ONE,14 ;ROUND UP
SACH   ANS,1
```

A common operation in DSP algorithms is the summation of products. The contents of the P register are added to the accumulator, and two values are simultaneously read and multiplied. A data memory value is multiplied by a program memory value. Example 5–32 shows multiplies and accumulates implemented with the LTA–MPY instruction pair.

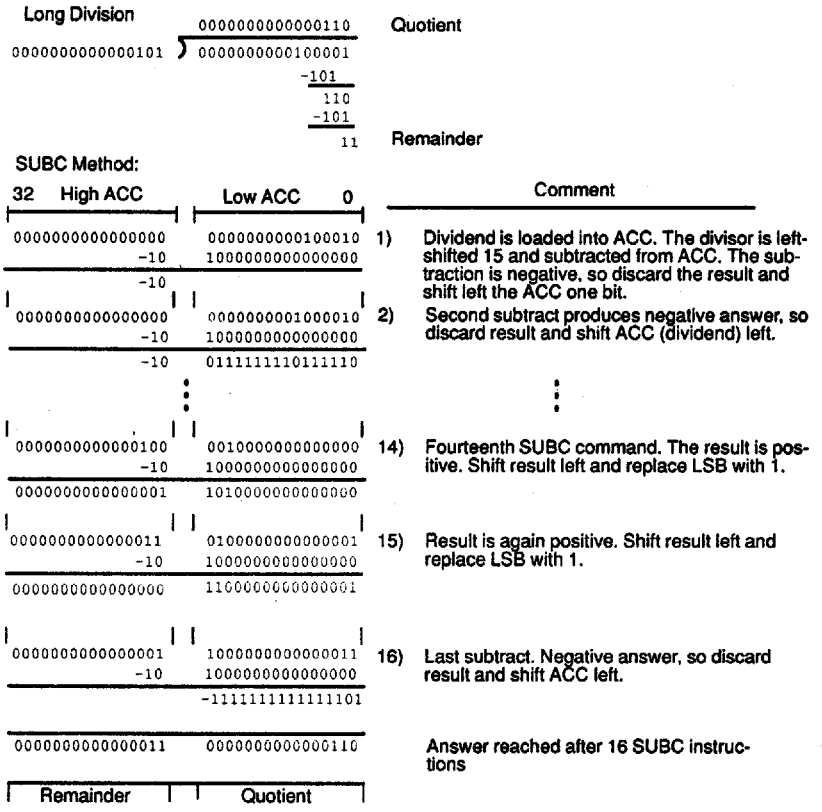
Example 5–32. Multiply and Accumulate Using the LTA–MPY Instruction Pair

```
*              CLOCK      TOTAL CLOCK   PROGRAM    TOTAL PROGRAM
*              CYCLES     CYCLES        MEMORY     MEMORY
*
    ZAC        1                        1
    LT     D1  1                        1
    MPY    C1  1                        1
    LTA    D2  1                        1
    MPY    C2  1                        1
    .
    .          2N                       2N
    .
    LTA    DN  1                        1
    MPY    CN  1                        1
    APAC       1          2 + 2N        1          2 + 2N
```

5.5.6 Division

Binary division is the inverse of multiplication. Multiplication consists of a series of shift and add operations, while division can be broken into a series of subtracts and shifts. Although the TMS320C1x does not have an explicit divide instruction, you can implement an efficient flexible divide capability with the conditional subtract instruction, SUBC. SUBC implements binary division like long division. Given a 16-bit positive dividend and divisor, the repetition of the SUBC command 16 times produces a 16-bit quotient in the low accumulator and a 16-bit remainder in the high accumulator. With each SUBC, the divisor is left-shifted 15 bits and subtracted from the accumulator. For each subtract that doesn't produce a negative answer, a one is put in the LSB of the quotient and then shifted. For each subtract that produces a negative answer, the accumulator is simply left-shifted. The shifting of the remainder and quotient after each subtract produces the separation of the quotient and remainder in the low and high halves of the accumulator. The similarities between long division and the SUBC method of division are shown in Figure 5–1, in which 33 is divided by 5.

Figure 5–1. Long Division and SUBC Division



The condition of the divisor, less than the shifted dividend, is determined by the sign of the result. The only restriction for the use of the SUBC instruction is that both the dividend and divisor *must* be positive. Thus, the sign of the quotient must be determined and the quotient computed by using the absolute value of the dividend and divisor. In addition, when you implement a divide algorithm, you must know if the quotient can be represented as a fraction and the degree of accuracy to which the quotient is to be computed. Each of these considerations can affect how the SUBC instruction is used (see Example 5–33 and Example 5–34). Note that the next instruction after SUBC cannot use the accumulator.

Example 5–33. Using SUBC With Numerator Smaller Than Denominator

```
* THIS ROUTINE DIVIDES TWO BINARY, 2S-COMPLEMENT
* NUMBERS OF ANY SIGN WHERE THE NUMERATOR IS LESS THAN
* THE DENOMINATOR.
*
*          BEFORE                  AFTER
*          INSTRUCTION             INSTRUCTION
*
* NUMERA   21                      21
* DENOM    42                      42
* QUOT     0                       0.5
*                                  (0.1 0 0)
*
*
DIV     LARP   0
        LT     NUMERA     ;GET SIGN OF QUOTIENT
        MPY    DENOM
        PAC
        SACH   TEMSGN     ;SAVE SIGN OF QUOTIENT
        LAC    DENOM
        ABS
        SACL   DENOM      ;MAKE DENOMINATOR POSITIVE
        ZALH   NUMERA     ;ALIGN NUMERATOR
        ABS               ;MAKE NUMERATOR POSITIVE
        LARK   0,14
*
* IF DIVISOR AND DIVIDEND ARE ALIGNED, DIVISION CAN START
* HERE.
*
KPDVNG SUBC    DENOM      ;15-CYCLE DIVIDE LOOP
        BANZ   KPDVNG
*
        SACL   QUOT
        LAC    TEMSGN
        BGEZ   DONE       ;DONE IF SIGN IS POSITIVE
        ZAC
        SUB    QUOT
        SACL   QUOT       ;NEGATE QUOTIENT IF NEGATIVE
DONE    RET               ;RETURN TO MAIN PROGRAM
```

Example 5–34. Using SUBC With Specified Quotient Accuracy

```
* THIS ROUTINE DIVIDES TWO BINARY, 2S-COMPLEMENT
* NUMBERS OF ANY SIGN, SPECIFYING THE FRACTIONAL
* ACCURACY OF THE QUOTIENT (FRAC).
*
*
*          BEFORE          AFTER
*          INSTRUCTION     INSTRUCTION
*
* NUMERA   11              11
* DENOM    8               8
* FRAC     3               3
* QUOT     17              1.375
*                          (1.0 1 1)
*
*
DN1     LT      NUMERA     ;GET SIGN OF QUOTIENT
        MPY     DENOM
        PAC
        SACH    TEMSGN     ;SAVE SIGN OF QUOTIENT
        LAC     DENOM
        ABS
        SACL    DENOM      ;MAKE DENOMINATOR POSITIVE
        LACK    15
        ADD     FRAC
        SACL    FRAC       ;COMPUTE LOOP COUNT
        LAC     NUMERA     ;ALIGN NUMERATOR
        ABS                ;MAKE NUMERATOR POSITIVE
        LAR     0,FRAC
*
* IF DIVISOR AND DIVIDEND ARE ALIGNED, DIVISION CAN START
* HERE.
*
KPDVNG  SUBC    DENOM      ;16 + FRAC CYCLE DIVIDE LOOP
        BANZ    KPDVNG
*
        SACL    QUOT
        LAC     TEMSGN
        BGEZ    DONE       ;DONE IF SIGN IS POSITIVE
        ZAC
        SUB     QUOT
        SACL    QUOT       ;NEGATE QUOTIENT IF NEGATIVE
DONE    RET                ;RETURN TO MAIN PROGRAM
```

5.5.7 Addition

In addition, both operands must be represented in the same Q format. Allow enough room in the result to accommodate bit growth, or prepare to handle overflows. If the operands are only 16 bits long, the result may have to be represented as a double-precision number. Example 5–35 and Example 5–36 illustrate two approaches to adding 16-bit numbers.

Example 5–35. Maintaining 32-Bit Results

```
        LAC     OP1        ;Q15
        ADD     OP2        ;Q15
        SACH    ANSHI      ;HIGH-ORDER 16 BITS OF RESULT
        SACL    ANSLO      ;LOW-ORDER 16 BITS OF RESULT
```

Example 5–36. Adjusted Binary Point to Maintain16-Bit Results

```
LAC    OP1,15     ;Q14 NUMBER IN ACCH
ADD    OP2,15     ;Q14 NUMBER IN ACCH
SACH   ANS        ;Q14
```

Double-precision operands present a more complex problem because actual arithmetic overflows or underflows may occur. The BV (branch on overflow) instruction can check for the occurrence of these conditions. A second technique is saturation mode operations, which will saturate the result of overflowing accumulations to the most positive or most negative number. Both techniques, however, result in a loss of precision. The best technique involves a thorough understanding of the underlying physical process and care in selecting number representations.

5.5.8 Floating-Point Arithmetic

Although the TMS320C1x devices are fixed-point 16/32-bit microprocessors, they can also perform floating-point computations. Using the floating-point single-precision standard proposed by the IEEE, the TMS320C1x can perform a floating-point multiplication in 8.4 μs and a floating-point addition in 17.2 μs. For a detailed discussion of floating-point arithmetic and TMS320 source code, refer to *Floating-Point Arithmetic with the TMS32010*, an application report in the book, *Digital Signal Processing Applications with the TMS320 Family, Volume 1* (literature number SPRA012A).

Floating-point numbers are often represented on microprocessors in a two-word format of mantissa and exponent. The mantissa is stored in one word. The exponent, the second word, indicates how many bit positions from the left the binary point is located. If the mantissa is 16 bits, a 4-bit exponent is sufficient to express the location of the binary point. Because of its 16-bit word size, the 16/4-bit floating-point format functions most efficiently on the TMS320C1x.

Operations in the TMS320C1x central ALU are performed in 2s-complement fixed-point notation. To implement floating-point arithmetic, operands must be converted to fixed point for arithmetic operations, and then converted back to floating point. Conversion to floating-point notation is performed by normalizing the input data (that is, shifting the MSB of the data word into the MSB of the internal memory word). The exponent word then indicates how many shifts are required. To multiply two floating-point numbers, multiply the mantissas and add the exponents. The resulting mantissa must be renormalized. (Because the input operands are normalized, not more than one left shift is required to normalize the result.)

Floating-point addition or subtraction requires shifting the mantissa so that the exponents of the two operands match. Use the difference between the exponents is to left-shift the lower power operand before adding. Then, renormalize the output of the add.

The LAC, LACK, ADD, and SUB instructions support floating-point operations. The mantissas are often used in Q15 format. Q format is a number representation common in operations on noninteger numbers. In Q format, the Q number (15 in Q15) denotes how many digits are located to the right of the binary point. A 16-bit number in Q15 format, therefore, has an assumed binary point immediately to the right of the most significant bit. Because the most significant bit constitutes the sign of the number, the numbers represented in Q15 may take on values from +1 (represented by +0.99997...) to −1.

5.6 Application-Oriented Operations

The TMS320C1x efficiently implements many common digital signal processing algorithms and provides solutions to numerically intensive problems usually characterized by multiply and accumulate operations. Some device-specific features that aid in the implementation of specific algorithms on the TMS320C1x include companding, filtering, fast Fourier transforms (FFT), and PID control. These applications require I/O performed either in parallel or in serial.

5.6.1 Companding

In the area of telecommunications, one of the primary concerns is the I/O bandwidth in the communications channel. One way to minimize this bandwidth is by companding (COMpress/exPAND). Companding is defined by two international standards, A-law and μ-law, both based on the compression of the equivalent of 13 bits of dynamic range into an 8-bit code. The standard employed in the United States and Japan is μ-law; the European standard is A-law. Detailed descriptions and code examples of μ-law and A-law companding are presented in *Companding Routines for the TMS32010/TMS32020*, an application report included in the book, *Digital Signal Processing Applications with the TMS320 Family, Volume 1* (literature number SPRA012A). The book also contains flowcharts, companding algorithms, and detailed descriptions). The algorithm space and time requirements for μ-law and A-law companding on the TMS320C10 and on the TMS320C15 are given in time.

The technique of companding allows the digital sample information corresponding to a 13-bit dynamic range to be transmitted as 8-bit data. For processing in the TMS320C1x, it is necessary to convert the 8-bit logarithmic data to a 16-bit linear format. Before outputting, the linear result must be converted to the compressed or companded format. On the TMS320C10 and TMS320C15, companding must be performed in software using conversion routines. On-chip companding hardware on the TMS320C17 implements these functions.

Table 5–3. Program Space and Time Requirements for μ-/A-Law Companding

	Words of Memory		Program Cycles		Time Req.‡
	Program	Data	Initialization	Loop†	μs
μ-Law:					
Compression	105	13	17	40	8.0
Expansion	46	8	6	23	4.6
A-Law:					
Compression	97	11	14	36	7.2
Expansion	48	7	4	25	5.0

† Worst case

‡ Assuming initialization

Four modes are available for the on-chip companding hardware operation on the TMS320C17: serial encode, serial decode, parallel encode, and parallel decode. The companding hardware converts between 2s-complement or sign-magnitude format and the companded format.

In the serial encode mode, transmitted data is encoded according to either μ-law or A-law format. In the serial decode mode, received data is decoded to a linear format according to the specified companding law.

In the parallel modes, either the encoder or decoder is enabled and then data written to port 1 is compressed or expanded. To convert sign-magnitude or 2s-complement linear PCM to 8-bit log PCM, the encoder is enabled for parallel operation, and the sample is written to port 1. An IN instruction from port 1 returns the linear PCM value. To convert an 8-bit log PCM to a sign-magnitude or 2s-complement linear PCM, the decoder is enabled for parallel operation, and the 8-bit sample is written to port 1. The expanded linear value is returned on the IN instruction from port 1. Note that when the selected conversion mode converts a 2s-complement value, there must be one instruction cycle between the OUT and IN instructions. Take care to have one OUT–IN instruction sequence to port 1 for each data sample because the execution of two OUT instructions to port 1 in succession pushes the first sample into the transmit register TR1, preventing access for read purposes. OUT instructions to port addresses 2 through 7 do not affect serial port operations.

When the companding hardware converts to sign-magnitude data, it must be converted to 2s-complement notation for computation in the microcomputer. Sign-magnitude notation consists of a sign bit in the MSB: a zero indicating a positive value, and a one indicating a negative number. All bits between the sign bit and the MSB of the data value are set to zero. For conversions between μ-law and sign-magnitude linear PCM, the hexadecimal value 1FFFh represents the most positive value of 8191, and the value 9FFFh represents the most negative value of –8191. For conversions between A-law and sign-magnitude linear PCM, the hexadecimal value 0FFFh represents the most positive value of 4095, and the value 8FFFh represents the most negative value of –4095.

Conversion between sign-magnitude and 2s-complement data for μ-law encoding and decoding is implemented with the code shown in Example 5–37 and Example 5–38, respectively. Conversion between 2s-complement and sign-magnitude data for A-law encoding and decoding is implemented with the code shown in Example 5–39 and Example 5–40, respectively. Note that both TMS320C17 devices feature hardware companding logic that can operate in either μ-law or A-law format with either sign-magnitude or 2s-complement numbers

Example 5–37. 2s-Complement to Sign-Magnitude for μ-Law Encoding

```
* THIS ROUTINE CONVERTS A 2S-COMPLEMENT NUMBER TO 14-BIT
* SIGN-MAGNITUDE FORMAT AND ADDS THE BIAS OF 33 FOR μ-LAW
* ENCODING. MEMORY LOCATION 1 CONTAINS THE VALUE 1 AND
* MEMORY LOCATION 2 (BIAS) CONTAINS +33.
*
OUTPUT
       LAC    SAMPLE     ;GET THE LINEAR DATA FOR OUTPUT
       BGEZ   POSOUT     ;IF POSITIVE, CHECK POS MAX VALUE
       ABS               ;IF NEGATIVE, CHECK ABSOIILUTE VALUE
       ADD    BIAS       ;ADD IN THE BIAS OF 21h
       ADD    ONE,15     ;SET THE SIGN BIT NEGATIVE
       SACL   SAMPLE     ;HOLD FOR LATER
       SUB    NEGMAX     ;COMPARE TO NEGATIVE MAX=9FFFh
       BLEZ   DONE       ;IF WITHIN MAX, THEN SEND IT
       LAC    NEGMAX     ;ELSE, LOAD THE VALUE WITH THE
       SACL   SAMPLE     ;LARGEST NEGATIVE IN RANGE
       B      DONE       ;AND SEND IT
POSOUT ADD    BIAS       ;ADD IN THE BIAS OF 21h
       SACL   SAMPLE     ;AND SAVE IT
       SUB    POSMAX     ;COMPARE TO POSITIVE MAX=1FFFh
       BLEZ   DONE       ;IF WITHIN MAX, THEN SEND IT
       LAC    POSMAX     ;ELSE, LOAD THE VALUE WITH THE
       SACL   SAMPLE     ;LARGEST POSITIVE VALUE IN RANGE
DONE   OUT    SAMPLE     ;PA1 AND SEND TO ENCODER
* ...  CONTINUE CODE HERE
```

Example 5–38. Sign-Magnitude to 2s-Complement for μ-Law Decoding

```
* THIS ROUTINE CONVERTS A 14-BIT SIGN-MAGNITUDE NUMBER TO
* 2S-COMPLEMENT NOTATION AND REMOVES THE BIAS OF 33 FOR
* μ-LAW DECODING. MEMORY LOCATION 1 CONTAINS THE VALUE 1
* AND MEMORY LOCATION 2 (BIAS) CONTAINS 33.
*
INPUT
       IN     SAMPLE,PA1;READ SERIAL PORT INPUT, DECODE
       LAC    SAMPLE     ;MOVE INPUT TO ACCUMULATOR
       SUB    BIAS       ;REMOVE BIAS VALUE
       BGEZ   POS        ;IF POSITIVE, THEN SAVE IT
       ADD    ONE,15     ;ELSE, DELETE SIGN BIT BY CARRY
       SACL   SAMPLE     ;SAVE MAGNITUDE VALUE
       ZAC               ;NEGATE THE INPUT BY
       SUB    SAMPLE     ;SUBTRACTING FROM ZERO AND SAVE
POS    SACL   SAMPLE     ;FULLY EXPANDED LINEAR DATA
* ...  CONTINUE CODE HERE
```

Example 5–39. 2s-Complement to Sign-Magnitude for A-Law Encoding

```
* THIS ROUTINE CONVERTS A 2S-COMPLEMENT NUMBER TO 13-BIT
* SIGN-MAGNITUDE NOTATION FOR A-LAW ENCODING. MEMORY
* LOCATION 1 CONTAINS THE VALUE 1.
*
OUTPUT
       LAC    SAMPLE     ;GET THE LINEAR DATA FOR OUTPUT
       BGEZ   POSOUT     ;IF POSITIVE, CHECK POS MAX VALUE
       ABS               ;IF NEGATIVE, CHECK NEG MAX VALUE
       ADD    ONE,15     ;SET THE SIGN BIT NEGATIVE
       SACL   SAMPLE     ;HOLD FOR LATER
       SUB    NEGMAX     ;COMPARE TO NEGATIVE MAX=8FFFh
       BLEZ   DONE       ;IF WITHIN MAX, THEN SEND IT
       LAC    NEGMAX     ;ELSE, LOAD THE VALUE WITH THE
       SACL   SAMPLE     ;LARGEST NEGATIVE IN RANGE
       B      DONE       ;AND SEND IT
POSOUT SACL   SAMPLE     ;SAVE IT
       SUB    POSMAX     ;COMPARE TO POSITIVE MAX=0FFFh
       BLEZ   DONE       ;IF WITHIN MAX, THEN SEND IT
       LAC    POSMAX     ;ELSE, LOAD THE VALUE WITH THE
       SACL   SAMPLE     ;LARGEST POSITIVE VALUE IN RANGE
DONE   OUT    SAMPLE,PA1;AND SEND IT TO  ENCODER
* ...  CONTINUE CODE HERE
```

Example 5–40. Sign-Magnitude to 2s-Complement for A-Law Decoding

```
* THIS ROUTINE CONVERTS A 13-BIT SIGN-MAGNITUDE NUMBER TO
* 2S-COMPLEMENT NOTATION FOR A-LAW ENCODING.
* MEMORY LOCATION 1 CONTAINS THE VALUE 1.
*
INPUT
       IN     SAMPLE,PA1;READ INPUT FROM SERIAL PORT; DECODE
       LAC    SAMPLE     ;MOVE INPUT TO ACCUMULATOR
       BGEZ   POS        ;IF POSITIVE, THEN SAVE IT
       ADD    ONE,15     ;ELSE, DELETE SIGN BIT BY CARRY
       SACL   SAMPLE     ;SAVE MAGNITUDE VALUE
       ZAC               ;NEGATE THE INPUT BY
       SUB    SAMPLE     ;SUBTRACTING FROM ZERO AND SAVE
       POS    SACL       ;SAMPLE, FULLY EXPANDED LINEAR DATA
* ...  CONTINUE CODE HERE
```

5.6.2 FIR/IIR Filtering

Digital filters are a common requirement for digital signal processing systems. The filters fall into two basic categories: finite impulse response (FIR) and infinite impulse response (IIR) filters. In either category, the coefficients of the filter (weighting factors) may be fixed or adapted during the course of the signal processing. The theory and implementation of digital filters has been presented and discussed in an application report, *Implementation of FIR/IIR Filters with the TMS32010/TMS32020*, included in the book, *Digital Signal Processing Applications with the TMS320 Family, Volume 1* (literature number SPRA012A).

IIR filters benefit from the fast instruction cycle time of the TMS320C1x. They typically require fewer multiply/accumulates. Correspondingly, the amount of

data memory for samples and coefficients is not usually a limiting factor. Because of sensitivity to quantization of the coefficients themselves, they are usually implemented in cascaded second-order sections. This translates to code consisting of LTD–MPY instruction pairs. Example 5–41 shows an implementation of a second-order IIR filter.

Example 5–41. Implementing an IIR Filter

```
* THE FOLLOWING EQUATIONS IMPLEMENT AN IIR FILTER:
*
* d(n) = x(n) + d(n-1)a1 + d(n-2)a2
* y(n) = d(n)b0 + d(n-1)b1 + d(n-2)b2
*
START  IN     XN,PA0     ;INPUT NEW VALUE XN
       LAC    XN,15      ;LOAD ACCUMULATOR WITH XN
*
       LT     DNM1
       MPY    A1
*
       LTD    DNM2
       MPY    A2
*
       APAC
       SACH   DN,1       ;d(n) = x(n) + d(n-1)a1 + d(n-2)a2
       ZAC
       MPY    B2
*
       LTD    DNM1
       MPY    B1
*
       LTD    DN
       MPY    B0
*
       APAC
       SACH   YN,1       ;y(n) = d(n)b0 + d(n-1)b1 + d(n-2)b2
       OUT    YN,PA1     ;YN IS THE OUTPUT OF THE FILTER
```

FIR filters, also, benefit from the fast instruction cycle time. An FIR filter requires many more multiply/accumulates than does the IIR filter with equivalent sharpness at the cutoff frequencies and with distortion and attenuation in the passbands and stopbands. The TMS320C1x devices help solve this problem by making implementation of longer filters feasible to implement. The TMS320C15 and the TMS320C17 have expanded data memory of 256 words, thus allowing additional coefficients and samples to be stored for longer length filters. Example 5–42 illustrates an implementation of a fourth-order (4 taps) FIR filter. Each tap consists of a LTD–MPY instruction pair, uses two data memory locations, and requires two instruction cycles to execute.

Example 5–42. Implementing an FIR Filter

```
* THE FOLLOWING EQUATION IS USED TO IMPLEMENT AN FIR FILTER:
*
* y(n)=[Ax(n-1)+Cx(n-3)+Dx(n-4)]* 2**-16
*
START  IN     X1,PA0     ;INPUT SAMPLE
       ZAC
*
       LT     X4         ;x(n-4)
       MPY    D
*
       LTD    X3         ;ACC=Dx4; x(n-4)=x(n-3)
       MPY    C
*
       LTD    X2         ;ACC=Dx4+Cx3; x(n-3)=x(n-2)
       MPY    B
*
       LTD    X1         ;ACC=Dx4+Cx3+Bx2; x(n-2)=x(n-1)
       MPY    A
*
       APAC              ;ACC=Dx4+Cx3+Bx2+Ax1
       SACH   Y,1
       OUT    Y,PA1      ;OUTPUT RESULTS
       B      START
```

Example 5–42 uses straightline code. For longer length FIR filters, straightline code may require larger program memory size. Depending on system constraints, you may choose to reduce program memory size by using looped code. However, straightline code runs much faster than looped versions. Consider the design tradeoff carefully.

5.6.3 Adaptive Filtering

With FIR or IIR filtering, the filter coefficients may be fixed or adapted. If the coefficients are adapted or updated with time, then another factor impacts the computational capacity. This factor is the requirement to adapt each of the coefficients, usually with each sample. A means of adapting the coefficients is the least-mean-square (LMS) algorithm given by the following equation:

$$b_k(i+1) = b_k(i) + 2Be(i)x(i-k)$$

where: $e(i) = x(i) - y(i)$

and: $$y(i) = \sum_{k=0}^{N-1} b_k x(i-k)$$

Quantization errors in the updated coefficients can be minimized if the result is obtained by rounding rather than truncating. For each coefficient in the filter at a given point in time, the factor 2B e(i) is a constant. This factor can then be computed once and stored in the T register for each of the updates. Thus, the computational requirement has become one multiply/accumulate plus rounding. The adaptation of each coefficient is five instructions corresponding to five clock cycles. This is shown in the instruction sequence as follows:

```
LARK   AR0,LASTAP    ;POINT TO DATA SAMPLE
LARK   AR1,COEFFD    ;POINT TO COEFFICIENTS
LARP   AR0
LT     ERRF          ;errf = 2B*e(i)
 .
 .
 .
MPY    *-,AR1        ;P = 2B*e(i)*X(i-0)
ZALH
APAC                 ;b0(i+1) = b0(1) + P
ADD    ONE,15        ;ROUND
SACH   *+,0,AR0      ;STORE b0(i+1)
.
.
.
```

Example 5–43 shows a routine to filter a signal and update the coefficients. The total execution time of the routine is 30 + 7n where n is the filter length. Data and program memory requirements are 5 + 2n words and 28 + 7n words, respectively. The filter length for adaptive filters is restricted by both execution time and memory. Obviously, more processing must be completed per sample because of the adaptation, and the size of the on-chip data RAM limits the number of coefficients and data samples that can be stored.

Another routine on adaptive filtering is discussed in the book, *Digital Signal Processing Applications with the TMS320 Family* (literature number SPRA012A); see application report, *Digital Voice Echo Canceller with a TMS32020.*

Example 5–43. 32-Tap Adaptive Filter

```
        .title    'ADAPTIVE FILTER'
        .def      ADPFIR
        .def      X,Y
*
* THIS 32-TAP ADAPTIVE FILTER USES PAGE 0 FOR COEFFICIENTS
* AND DATA SAMPLES. THE NEWEST INPUT SHOULD BE IN MEMORY
* LOCATION X WHEN CALLED. THE OUTPUT WILL BE IN MEMORY
* LOCATION Y WHEN RETURNED.
```

```
*
ONE     .set    120         ;CONSTANT ONE
BETA    .set    121         ;ADAPTATION CONSTANT * 2
ERR     .set    122         ;SIGNAL ERROR
ERRF    .set    123         ;ERROR FUNCTION
Y       .set    124         ;FILTER OUTPUT
X       .set    125         ;NEWEST DATA SAMPLE
FRSTAT .set     32          ;NEXT NEWEST DATA SAMPLE
LASTAP .set     63          ;OLDEST DATA SAMPLE
COEFFD .set     0           ;START OF COERRICIENT TABLE
*
* FINITE IMPULSE RESPONSE (FIR) FILTER.
*
        .text
ADPFIR LDPK     0           ;USE DATA PAGE 0
       LARK     AR1,COEFFD;LOAD POINTER FOR COEFF TABLE
       LARK     AR0,LASTAP;LOAD POINTER FOR DATA SAMPLES
       MPYK     0           ;CLEAR THE P REGISTER
       LAC      ONE,14      ;LOAD OUTPUT ROUNDING BIT
       LARP     AR0
*
* DO 32 TAPS.
*
FIR    LT       *-,AR1      ;LOAD T REG WITH OLDEST SAMPLE
       MPY      *+,AR0      ;MULTIPLY WITH LAST COEFFICIENT
*
*
       LTD      *-,AR1      ;LOAD NEXT SAMPLE
       MPY      *+,AR0      ;MULTIPLY WITH NEXT COEFFICIENT
*
       LTD      *-,AR1      ;LOAD NEXT SAMPLE
       MPY      *+,AR0      ;MULTIPLY WITH NEXT COEFFICIENT
         .
         .
         .
       LTD      *-,AR1      ;LOAD LAST SAMPLE
       MPY      *+,AR0      ;MULTIPLY WITH LAST COEFFICIENT
*
       APAC
       SACH     Y,1         ;STORE FILTER OUTPUT
       ZAC
       SUB      Y           ;ACC = -y(i)
       ADD      X           ;ADD THE NEWEST INPUT
       SACL     ERR         ;err(i) = x(i) - y(i)
*
* LMS ADAPTATION OF FILTER COEFFICIENTS.
*
       LT       ERR
       MPY      BETA
       PAC                  ;errf(i) = 2*beta*err(i)
       ADD      ONE,14      ;ROUND THE RESULT
       SACH     ERRF,1
       LAC      X
       SACL     FRSTAP      ;INCLUDE NEWEST SAMPLE
*
       LARK     AR0,LASTAP;POINT TO DATA SAMPLE
       LARK     AR1,COEFFD;POINT TO COEFFICIENTS
*
       LT       ERRF        ;KEEP ERRF IN T REGISTER
```

```
*
ADAPT  MPY    *-,AR1     ;P = 2*beta*err(i)*x(i-31)
       ZALH   *
       APAC              ;b31(i+1) = b31(i) + P
       ADD    ONE,15     ;ROUND
       SACH   *+,0,AR0   ;STORE b31(i+1)
*
       MPY    *-,AR1     ;P = 2*beta*err(i)*x(i-30)
       ZALH   *
       APAC              ;b30(i+1) = b30(i) + P
       ADD    ONE,15     ;ROUND
       SACH   *+,0,AR0   ;STORE b30(i+1)
*
       MPY    *-,AR1     ;P = 2*beta*err(i)*x(i-29)
       ZALH   *
       APAC              ;b29(i+1) = b29(i) + P
       ADD    ONE,15     ;ROUND
       SACH   *+,0,AR0   ;STORE b29(i+1)
         .
         .
         .
       MPY    *-,AR1     ;P = 2*beta*err(i)*x(i-0)
       ZALH   *
       APAC              ;b0(i+1) = b0(i) + P
       ADD    ONE,15     ;ROUND
       SACH   *+,0,AR0   ;STORE b0(i+1)
*
       RET               ;RETURN TO MAIN PROGRAM
```

5.6.4 Fast Fourier Transforms (FFT)

Fourier transforms are another important tool often used in digital signal processing systems. The purpose of the transform is to convert information from the time domain to the frequency domain. The inverse Fourier transform converts information back to the time domain from the frequency domain. Implementations of Fourier transforms that are computationally efficient are known as fast Fourier transforms (FFTs). The theory and implementation of FFTs has been discussed in the book, *DFT/FFT and Convolution Algorithms*, by Burrus and Parks, published by John Wiley and Sons. The book also contains a large number of sample TMS32010 and FORTRAN programs to implement DFT/FFT algorithms. The TMS320C1x reduces the execution time of all FFTs by virtue of its single-cycle instruction time.

Example 5–44 consists of some of the macros used in the implementation of FFTs. Example 5–45 shows the code for an 8-point DIT (decimation in time) FFT. The code has been structured into a number of macro calls, including a macro for bit reversal.

Example 5–44. FFT Macros

```
COMBO     .macro R1,I1,R2,I2,R3,I3,R4,I4
*
* CALCULATE PARTIAL TERMS FOR R3, R4, I3, AND I4.
*
      LAC    :R3:,14   ;ACC  :=   (1/4)(R3)
      ADD    :R4:,14   ;ACC  :=   (1/4)(R3+R4)
      SACH   :R3:,1    ;R3   :=   (1/2)(R3+R4)
      SUB    :R4:,15   ;AC   :=   (1/4)(R3+R4)-(1/2)(R4)
      SACH   :R4:,1    ;R4   :=   (1/2)(R3-R4)
      LAC    :I3:,14   ;ACC  :=   (1/4)(I3)
      ADD    :I4:,14   ;ACC  :=   (1/4)(I3+I4)
      SACH   :I3:,1    ;I3   :=   (1/2)(I3+I4)
      SUB    :I4:,15   ;ACC  :=   (1/4)(I3+I4)-(1/2)(I4)
      SACH   :I4:,1    ;I4   :=   (1/2)(I3-I4)
*
* CALCULATE PARTIAL TERMS FOR R2, R4, I2, AND I4.
*
      LAC    :R1:,14   ;ACC  :=   (1/4)(R1)
      ADD    :R2:,14   ;ACC  :=   (1/4)(R1+R2)
      SACH   :R1:,1    ;R1   :=   (1/2)(R1+R2)
      SUB    :I3:,15   ;AC   :=   (1/4)(R1+R2)-(1/2)(R2)
      ADD    :I4:,15   ;ACC  :=   (1/4)[(R1-R2)+(I3-I4)]
      SACH   :R2:      ;R2   :=   (1/4)[(R1-R2)+(I3-I4)]
      SUBH   :I4:      ;ACC  :=   (1/4)[(R1-R2)-(I3-I4)]
      DMOV   :R4:      ;I4   :=  R4 = (1/2)(R3-R4)
      SACH   :R4:      ;R4   :=   (1/4)[(R1-R2)-(I3-I4)]
      LAC    :I1:,14   ;ACC  :=   (1/4)(I1)
      ADD    :I2:,14   ;ACC  :=   (1/4)(I1+I2)
      SACH   :I1:,1    ;I1   :=   (1/2)(I1+I2)
      SUB    :I2:,15   ;ACC  :=   (1/4)(I1+I2)-(1/2)(I2)
      SUB    :I4:,15   ;ACC  :=   (1/4)[(I1-I2)-(I3-I4)]
      SACH   :I2:      ;I2   :=   (1/4)[(I1-I2)-(I3-I4)]
      ADDH   :I4:      ;ACC  :=   (1/4)[(I1-I2)+(I3-I4)]
      SACH   :I4:      ;I4   :=   (1/4)[(I1-I2)+(I3-I4)]
*
* CALCULATE PARTIAL TERMS FOR R1, R3, I1, AND I3.
*
      LAC    :R1:,15   ;ACC  :=   (1/4)(R1+R2)
      ADD    :R3:,15   ;ACC  :=   (1/4)[(R1+R2)+(R3+R4)]
      SACH   :R1:      ;R1   :=   (1/4)[(R1+R2)+(R3+R4)]
      SUBH   :R3:      ;ACC  :=   (1/4)[(R1+R2)-(R3+R4)]
      SACH   :R3:      ;R3   :=   (1/4)[(R1+R2)-(R3+R4)]
      LAC    :I1:,15   ;ACC  :=   (1/4)(I1+I2)
      ADD    :I3:,15   ;ACC  :=   (1/4)[(I1+I2)+(I3+I4)]
      SACH   :I1:      ;I1   :=   (1/4)[(I1+I2)+(I3+I4)]
      SUBH   :I3:      ;ACC  :=   (1/4)[(I1+I2)-(I3+I4)]
      SACH   :I3:      ;I3   :=   (1/4)[(I1+I2)-(I3+I4)]
       .endm
```

```
*
* MACRO FOR INPUT BIT REVERSAL.
*
BITREV .macro PR,PI,QR,QI
       ZALH   :PR:
       ADDS   :QR:
       SACL   :PR:
       SACH   :QR:
       ZALH   :PI:
       ADDS   :QI:
       SACL   :PI:
       SACH   :QI:
       .endm
*
ZERO   .macro PR,PI,QR,QI
*
* CALCULATE Re(P+Q) AND Re(P-Q)
*
       LAC    :PR:,15       ;ACC   := (1/2)(PR)
       ADD    :QR:,15       ;ACC   := (1/2)(PR+QR)
       SACH   :PR:          ;PR    := (1/2)(PR+QR)
       SUBH   :QR:          ;ACC   := (1/2)(PR+QR)-(QR)
       SACH   :QR:          ;QR    := (1/2)(PR-QR)
*
* CALCULATE Im(P+Q) AND Im(P-Q)
*
       LAC    :PI:,15       ;ACC   := (1/2)(PI)
       ADD    :QI:,15       ;ACC   := (1/2)(PI+QI)
       SACH   :PI:          ;PR    := (1/2)(PI+QI)
       SUBH   :QI:          ;ACC   := (1/2)(PI+QI)-(QI)
       SACH   :QI:          ;QR    := (1/2)(PI-QI)
       .endm
*
*
PIBY4  .macro PR,PI,QR,QI,W
*
       LT     :W:           ;T REG := W=COS(PI/4)=SIN(PI/4)
       LAC    :QI:,14       ;ACC   := (1/4)(QI)
       SUB    :QR:,14       ;ACC   := (1/4)(QI-QR)
       SACH   :QI:,1        ;QI    := (1/2)(QI-QR)
       ADD    :QR:,15       ;ACC   := (1/4)(QI+QR)
       SACH   :QR:,1        ;QR    := (1/2)(QI+QR)
       LAC    :PR:,14       ;ACC   := (1/4)(PR)
       MPY    :QR:          ;P REG := (1/4)(QI+QR)*W
       APAC                 ;ACC   := 1/4)[PR+(QI+QR)*W]
       SACH   :PR:,1        ;PR    := (1/2)[PR+(QI+QR)*W]
       SPAC                 ;ACC   := (1/4)(PR)
       SPAC                 ;ACC   := (1/4)[PR-(QI+QR)*W]
       SACH   :QR:,1        ;QR    := (1/2)[PR-(QI+QR)*W]
       LAC    :PI:,14       ;ACC   := (1/4)(PI)
       MPY    :QI:          ;P REG := (1/4)(QI-QR)*W
       APAC                 ;ACC   := (1/4)[PI+(QI-QR)*W]
       SACH   :PI:,1        ;PI    := (1/2)[PI+(QI-QR)*W]
       SPAC                 ;ACC   := (1/4)(PI)
       SPAC                 ;ACC   := (1/4)[PI-(QI-QR)*W]
       SACH   :QI:,1        ;QI    := (1/2)[PI-(QI-QR)*W]
       .endm
```

```
*
*
PIBY2  .macro PR,PI,QR,QI
*
* CALCULATE   Re(P+jQ) AND Re(P-jQ)
*
       LAC    :PI:,15       ;ACC   := (1/2)(PI)
       SUB    :QR:,15       ;ACC   := (1/2)(PI-QR)
       SACH   :PI:          ;PI    := (1/2)(PI-QR)
       ADDH   :QR:          ;ACC   := (1/2)(PI-QR)+(QR)
       SACH   :QR:          ;QR    := (1/2)(PI+QR)
*
* CALCULATE   Im(P+jQ) AND Im(P-jQ)
*
       LAC    :PR:,15       ;ACC   := (1/2)(PR)
       ADD    :QI:,15       ;ACC   := (1/2)(PR+QI)
       SACH   :PR:          ;PR    := (1/2)(PR+QI)
       SUBH   :QI           ;ACC   := (1/2)(PR+QI)-(QI)
       DMOV   :QR:          ;QR --> QI
       SACH   :QR:          ;QR    := (1/2)(PR-QI)
       .endm
*
*
PI3BY4 .macro    PR,PI,QR,QI,W
*
       LT     :W:           ;T REG := W=COS(PI/4)=SIN(PI/4)
       LAC    :QI:,14       ;ACC   := (1/4)(QI)
       SUB    :QR:,14       ;ACC   := (1/4)(QI-QR)
       SACH   :QI:,1        ;QI    := (1/2)(QI-QR)
       ADD    :QR:,15       ;ACC   := (1/4)(QI+QR)
       SACH   :QR:,1        ;QR    := (1/2)(QI+QR)
       LAC    :PR:,14       ;ACC   := (1/4)(PR)
       MPY    :QI:          ;P REG := (1/4)(QI-QR)*W
       APAC                 ;ACC   := (1/4)[PR+(QI-QR)*W]
       SACH   :PR:,1        ;PR    := (1/2)[PR+(QI-QR)*W]
       SPAC                 ;ACC   := (1/4)(PR)
       SPAC                 ;ACC   := (1/4)[PR-(QI-QR)*W]
       MPY    :QR:          ;P REG := (1/4)(QI+QR)*W
       SACH   :QR:,1        ;QR    := (1/2)[PR-(QI-QR)*W]
       LAC    :PI:,14       ;ACC   := (1/4)(PI)
       SPAC                 ;ACC   := (1/4)[PI-(QI+QR)*W]
       SACH   :PI:,1        ;PI    := (1/2)[PI-(QI+QR)*W]
       APAC                 ;ACC   := (1/4)(PI)
       APAC                 ;ACC   := (1/4)[PI+(QI+QR)*W]
       SACH   :QI:,1        ;QI    := (1/2)[PI+(QI+QR)*W]
       .endm
```

Example 5–45. An 8-Point DIT FFT

```
* THIS ROUTINE IMPLEMENTS AN 8-POINT DIT FFT. ASSUME THAT
* TWIDDLE FACTOR = W VALUE STORED IN MEMORY LOCATION W.
*
*
XOR     .set   00
XOI     .set   01
X1R     .set   02
X1I     .set   03
X2R     .set   04
X2I     .set   05
X3R     .set   06
X3I     .set   07
X4R     .set   08
X4I     .set   09
X5R     .set   10
X5I     .set   11
X6R     .set   12
X6I     .set   13
X7R     .set   14
X7I     .set   15
W       .set   16
WVALUE .set    5A82h  ;VALUE FOR SIN (45) OR COS (45)
*
* INITIALIZE FFT PROCESSING. ASSUME TWIDDLE FACTOR =
* W VALUE STORED IN MEMORY LOCATION W.
*
        .text
FFT     ROVM           ;RESET OVERFLOW MODE
        LDPK   0       ;SET DATA PAGE POINTER TO 0
*
*   BIT-REVERSED INPUT SAMPLES.
*
        BITREV X1R,X1I,X4R,X4I
        BITREV X3R,X3I,X6R,X6I
*
* FIRST AND SECOND STAGES COMBINED WITH DIVIDE-BY-4
* INTERSTAGE SCALING.
*
        COMBO  X0R,X0I,X1R,X1I,X2R,X2I,X3R,X3I,
        COMBO  X4R,X4I,X5R,X5I,X6R,X6I,X7R,X7I.
*
* THIRD STAGE WITH DIVIDE-BY-2 INTERSTAGE SCALING.
*
        ZERO   X0R,X0I,X4R,X4I
        PIBY4  X1R,X1I,X5R,X5I,W
        PIBY2  X2R,X2I,X6R,X6I
        PI3BY4 X3R,X3I,X7R,X7I,W
*
```

5.6.5 PID Control

The purpose of control systems is to regulate a process and achieve a desired behavior or output from the process. A control system consists of three main components: sensors, actuators, and a controller. Sensors measure the behavior of the system. Actuators supply the driving force to ensure the desired behavior. The controller generates actuator commands corresponding to the error conditions observed by the sensors and the control algorithms programmed in the controller. The controller typically consists of an analog or digital processor.

Analog control systems are usually based on fixed components and are not programmable. They are also limited to using single-purpose characteristics of the error signal, such as P (proportional), I (integral), and D (derivative), or their combination. These limitations, along with other disadvantages of analog systems such as component aging and temperature drift, are reasons why digital control systems increasingly replace analog systems in most control applications.

Digital control systems that use a microprocessor/microcontroller are able to implement more sophisticated algorithms of modern control theory, such as state models, deadbeat control, state estimation, optimal control, and adaptive control. Digital control algorithms deal with the processing of digital signals and are similar to DSP algorithms.

The most commonly used algorithm in both analog and digital control systems is the PID (proportional, integral, and derivative) algorithm. The classical PID algorithm is given by

$$u(t) = K_p e(t) + K_i \int e\,dt + K_d \frac{de}{dt}$$

The PID algorithm must be converted into a digital form for implementation on a microprocessor. Using a rectangular approximation for the integral, the PID algorithm can be approximated as

$$u(n) = u(n-1) + K_1\, e(n) + K_2\, e(n-1) + K_3\, e(n-2)$$

This algorithm is implemented in Example 5–46.

Example 5–46. PID Control

```
        .title 'PID CONTROL'
        .def   PID
*
* THIS ROUTINE IMPLEMENTS A PID ALGORITHM.
*
UN      .set   0        ;OUTPUT OF CONTROLLER
E0      .set   1        ;LATEST ERROR SAMPLE
E1      .set   2        ;PREVIOUS ERROR SAMPPLE

*
UN      .set   0        ;OUTPUT OF CONTROLLER
E0      .set   1        ;LATEST ERROR SAMPLE
E1      .set   2        ;PREVIOUS ERROR SAMPLE
E2      .set   3        ;OLDEST ERROR SAMPLE
K1      .set   4        ;GAIN CONSTANT
K2      .set   5        ;GAIN CONSTANT
K3      .set   6        ;GAIN CONSTANT
*
* ASSUME DATA PAGE 0 IS SELECTED.
*
        .text
        PID    IN  E0,PA0 ;READ NEW ERROR SAMPLE
        LAC    UN         ;ACC = u(n-1)
        LT     E2         ;LOAD T REG WITH OLDEST SAMPLE
        MPY    K2         ;P = K2*e(n-2)
        LTD    E1         ;ACC = u(n-1)+K2*e(n-2)
        MPY    K1         ;P = K1*e(n-1)
        LTD    E0         ;ACC = u(n-1)+K1*e(n-1)+K2*e(n-2)
        MPY    K0         ;P = K0*e(n)
        APAC              ;ACC = u(n-1)+K0*e(n)+K1*e(n-1)
*                         ;+K2*e(n-2)
        SACH   UN,1       ;STORE OUTPUT
        OUT    UN,PA1     ;SEND IT
```

The PID loop executes in 13 cycles or 2.6 µs at a 20-MHz clock rate. The TMS320 can also be used to implement more sophisticated algorithms such as state modeling, adaptive control, state estimation, Kalman filtering, and optimal control. Other functions that can be implemented are noise filtering, stability analysis, and additional control loops.

5.6.6 Self-Test Routines

A self-test program can effectively perform incoming quality verification or be used as a powerup device verification tool. Texas Instruments has developed a self-test program to check out the functionality of a TMS320C1x device before you branch to the user code. This program is not intended to provide a means of logic debug but rather to indicate device pass/fail from which it can be determined whether or not the TMS320C1x is functional.

When designing a DSP device, Texas Instruments runs thorough patterns through the logic to test all the stages. In these patterns, worst-case conditions and transitions are forced to verify logic design prior to manufacturing. Likewise, the speed and electrical specifications are thoroughly tested. In produc-

tion manufacturing, every TMS320C1x is tested to meet the functionality, speed, and power specifications of the device before it is shipped. The drive levels and loading of lines are checked at full speed and over varying temperature.

The 460-word self-test program for the TMS320C1x exercises most of the on-chip resources of the device with a minimal amount of external circuitry. Note that this code is intended for testing on-chip resources and will not exercise the external interface lines.

Example 5–47 contains a small portion of this self-test program, which checks out the ALU section. The ALU test is designed to validate the basic operation of the circuit. It consists of a series of subtests to verify addition and subtraction operations of both halves of the 32-bit operation as well as carry and overflow calculations, absolute value, and SUBC operation. A failure in any of these tests will set the error code in the accumulator to 100xh, where x is the number of the subtest that has failed.

Other sections of this self-test check the auxiliary registers, on-chip data RAM, on-chip program ROM (longitudinal redundancy test), status register and branches, pre- and post-scaling shifters, multiplier, and the instruction set.

Texas Instruments offers an applications brief, which discusses the code segments that compose the TMS320C1x self-test program and also explains how to link and execute this code. The applications brief and selftest code are available through the TMS320 DSP Bulletin Board Service (see Appendix E).

Example 5–47. Self-Test Routine

```
* THIS PROGRAM EXECUTES AN INTERNAL SELFTEST OF THE
* TMS320C1X MICROCOMPUTER ALU. A FAILURE IN ANY OF THESE
* TESTS WILL SET THE ERROR CODE IN THE ACCUMULATOR TO 100Xh
* WHERE X IS THE NUMBER OF THE SELFTEST THAT HAS FAILED.

*
*
* RESET AND INTERRUPT VECTORS.
*
BEGIN  B      START     ;RESET SOFT VECTOR
       B      INTRPT    ;INTERRUPT SOFT VECTOR
*
* REQUIRED DATA VALUES FOR TEST PROGRAMS.
*
       .word  0FFFFh    ;RAM TEST PATTERN 1
       .word  0AAAAh    ;RAM TEST PATTERN 2
       .word  5555h     ;RAM TEST PATTERN 3
       .word  0h        ;RAM TEST PATTERN 4
*
* PROGRAM INITIALIZATION DP = 0 AND DISABLE INTERRUPTS.
*
       .text
START                   ;START INITIALIZATION ROUTINE
       LDPK   0         ;START IN ZERO DATA PAGE
       DINT             ;DISABLE EXTERNAL INTERRUPTS
*
* ARITHMETIC LOGIC UNIT TEST.
*
ALU
       LACK   1         ;GET INCREMENT VALUE
       SACL   8         ;STORE IT IN REG8
       LACK   4         ;POINT ACC TO PATTERNS TABLE
       TBLR   4         ;PUT TABLE VALUE IN REG4
       ADD    8         ;INCREMENT TABLE ADDRESS
       TBLR   5         ;PUT TABLE VALUE IN REG5
       ADD    8         ;INCREMENT TABLE ADDRESS
       TBLR   6         ;PUT TABLE VALUE IN REG6
       ADD    8         ;INCREMENT TABLE ADDRESS
       TBLR   7         ;PUT TABLE VALUE IN REG7
       LACK   10h       ;SET ERROR CODE VALUE
       SACL   2         ;STORE CODE IN REG2
*
```

```
ALU1    ZAC             ;CLEAR OUT ACCUMULATOR
        ADDS    5       ;ADD IN 0AAAAh PATTERN
        AND     5       ;AND WITH 0AAAAh PATTERN
        OR      6       ;OR WITH 5555h PATTERN
        SUBS    4       ;SUBTRACT -1 FROM PATTERN
        BZ      ALU2    ;IF ACC CLEARED, GO TO NEXT TEST
*
        LACK    1       ;IF NOT, THEN SET TEST 1 CODE
        ADD     2,8     ;ADD IN ERROR CODE
        B       ERROR   ;EXIT TO ERROR ROUTINE
*
ALU2    ZALH    5       ;ADD HIGH THE 0AAAAh PATTERN
        ADDH    6       ;SUBTRACT HIGH THE 5555h PATTERN
        SACH    0       ;SAVE THE VALUE
        ZALH    0       ;RESTORE THE VALUE
        ABS             ;TAKE ABSOLUTE VALUE
        SUBH    8       ;SUBTRACT HIGH 10000h
        BZ      ALU3    ;IF ACC CLEARED, GO TO NEXT TEST
*
        LACK    2       ;IF NOT, THEN SET TEST 2 CODE
        ADD     2,8     ;ADD IN ERROR CODE
        B       ERROR   ;EXIT TO ERROR ROUTINE
*
ALU3    LAC     4,12    ;LOAD ACC WITH 0FFFFF000h PATTERN
        ADD     8,12    ;ADD 00001000h TO IT
        BZ      ALU4    ;IF ACC CLEARED, GO TO NEXT TEST
*
        LACK    3       ;IF NOT, THEN SET TEST 3 CODE
        ADD     2,8     ;ADD IN ERROR CODE
        B       ERROR   ;EXIT TO ERROR ROUTINE
*
ALU4    ADD     4       ;LOAD ACC WITH 0FFFFFFFFh PATTERN
        ABS             ;TAKE ABSOLUTE VALUE
        SUB     8       ;SUBTRACT 00000001h
        BZ      ALU5    ;IF ACC CLEARED, GO TO NEXT TEST
*
        LACK    4       ;IF NOT, THEN SET TEST 4 CODE
        ADD     2,8     ;ADD IN ERROR CODE
        B       ERROR   ;EXIT TO ERROR ROUTINE
*
ALU5    LACK    40h     ;GET DIVISOR = 64
        SACL    0       ;SAVE IN REG0
        LACK    0FFh    ;GET DIVIDEND = 255
        SUBC    0       ;1ST STAGE OF DIVIDE
        NOP             ;REQUIRED NOP
        SUBC    0       ;2ND STAGE OF DIVIDE
        NOP             ;REQUIRED NOP
          .
          .
          .
        SUBC    0       ;16TH STAGE OF DIVIDE
        NOP             ;REQUIRED NOP
        SACH    1       ;SAVE REMAINDER
        SACL    2       ;SAVE QUOTIENT
        LACK            ;GET QUOTIENT COMPARISON MASK
        XOR     2       ;COMPARE WITH CALCULATED ANSWER
        BZ      ALU6    ;IF ACC CLEARED, GO TO NEXT TEST
```

```
*
       LACK   5              ;IF NOT, THEN SET TEST 5 CODE
       ADD    2,8            ;ADD IN ERROR CODE
       B      ERROR          ;EXIT TO ERROR ROUTINE
*
ALU6   LACK   3Fh            ;GET REMAINDER COMPARISON MASK
       XOR    1              ;COMPARE WITH ANSWER
       BZ     STATUS         ;IF ACC CLEARED, GO TO NEXT TEST
*
       LACK   6              ;IF NOT, THEN SET TEST 6 CODE
       ADD    2,8            ;ADD IN ERROR CODE
       B      ERROR          ;EXIT TO ERROR ROUTINE
```

5.7 PWM Generation (TMS320C14)

The TMS320C14 can be configured to generate high-precision pulse width modulation (PWM) outputs. In this function, the compare output pins are controlled directly by the compare registers. The contents of the compare registers control the duration of the high portion of the PWM waveform. The PWM output is useful in controlling stepper motors.

Example 5–48 contains code that could be used in adjusting the phase response of a three-phase motor. By controlling the width of the PWM waveform output, a designer can control the relative phase of a three-phase motor. Timer 2 is used to determine the duration of the PWM pulse width.

Example 5–48. Using Compare Outputs for Motor Control

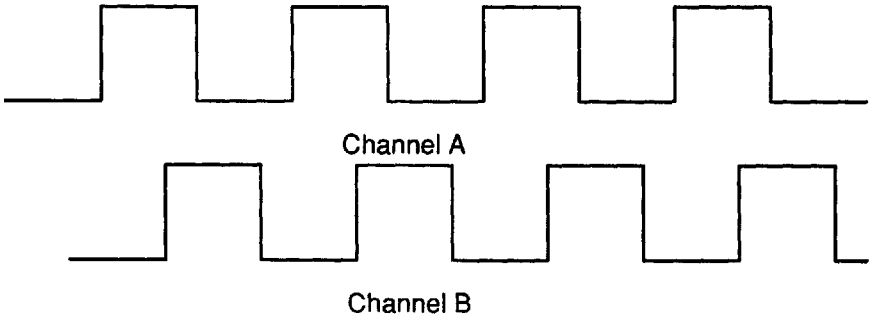
```
*
* PWM GENERATION WITH COMPARE OUTPUTS EXAMPLE. SETTING BIT 8
* OF THE TCON REGISTER OVERRIDES ANY OTHER CONFIGURATION. THE
* COMPARE REGISTER(S) MUST BE SET UP WITH THE DURATION OF THE
* HIGH PORTION OF THE PWM OUTPUT. IF THESE VALUES ARE TO BE
* CHANGED, WRITING TO THE PROPER ACTION REGISTER ACCOMPLISHES
* THE TASK IMMEDIATELY AT THE END OF THE CURRENT CYCLE.
*
      .include  "C14INC"  ;INCLUDE HEADER FILE
*
* IT IS ASSUMED THAT'BANK' AND 'TMP' ARE AVAILABLE TO STORE
* VALUES TO THE BSR AND THE CONTROL REGISTER WHICH IS TO BE
* ALTERED. IT IS ALSO ASSUMED THAT 'ONE' HOLDS THE CONSTANT 1.
*
      .ref      BANK,TMP,ONE
*
* IT IS ALSO ASSUMED THAT 'PWMperiod' HOLDS THE DESIRED PERIOD
* AND 'PWMwidth' HOLDS THE WIDTH.
*
      .ref      PWMperiod,PWMwidth
*
*     .
*     .
*     .
      .data
*
TABLE:
      .word     CMPPWMMode+CMPTMR1Select+TMR1Enable*TMRby1+TMR1Internal
*     .
*     .
*     .
      .text
*
* SET UP PERIOD REGISTER OF TIMER 1
*
      LACK      TimerBank
      SACL      BANK
      OUT       BANK,BSR
      OUT       PWMperiod,TPR1
*
* GET MODE WORD FOR TCON
*
      LT        ONE
      MPYK      TABLE
      PAC
      TBLR      TMP
      OUT       TMP,TCON      ;LOAD TCON
*
      LACK      ActionBank
      SACL      BANK
      OUT       BANK,BSR
      OUT       PWMwidth,ACT5;UPDATE PWM WIDTH
*
*     .
*     .
*     .
      .end
```

5.8 Speed/Position Measurement (TMS320C14)

The capture subsystem on the TMS320C14 is capable of detecting and storing any transition on its capture input pins. Using optical encoders, a digital representation of an analog process such as speed measurement can be obtained. An incremental encoder creates a series of square waves. The number of square waves corresponds to the mechanical increment required. For an encoder that supplies 1024 square wave cycles per revolution (360 degrees), each increment of one corresponds to 0.351 degrees of revolution. Using a counter, the rotation of the shaft can be calculated.

A dual channel incremental encoder can be used for position sensing, as shown in Figure 5–2. Most incremental systems use two output channels in quadrature so that you can count the transitions from a high state to a low state, and view the state of the opposite channel during these transitions. Using this information, you can determine if channel A leads channel B and derive the direction.

Figure 5–2. Dual-Channel Optical Encoder Outputs



Example 5–49 instructs the CPU to read the capture input once every 0.5 ms, compare the input to a value (contents of Timer 1), and toggle the bit I/O pins 0 and 1 for every match.

Example 5–49. Using the Capture Inputs to Detect Speed

```
*
* USE CAPTURE INPUTS TO DETERMINE SPEED. BITS 11 THROUGH 15
* OF TCON ARE PRESUMED TO BE 0. THIS THEN IMPLIES THAT TIMER 1
* IS USED FOR CAPTURE OPERATIONS WITH INTERNAL CLOCK SOURCE
* WITH AN INTERRUPT GENERATED ON THE FIRST CAPTURE ENTRY
* RECEIVED IN FIFO0. THE OPTICAL ENCODER OUTPUTS A AND B ARE
* CONNECTED TO CAP0 AND CAP1 INPUTS. ANY TRANSITION FROM HIGH
* TO LOW WILL TRIGGER A CAPTURE. THIS ROUTINE COMPUTES MEAN
* TIME BETWEEN A AND B PULSES AND DETERMINES SIGN OF FORWARD
* DIFFERENCE OF TIME BETWEEN AN A PULSE AND A B PULSE.
*
* INCLUDE C14 EQUATES
*
```

```
        .include  "C14INC"          ;INCLUDE HEADER FILE
*
        .def      CaptureInit,CaptureISR
*
        .data
*
TABLE:
        .word  CAP0Mode*NegEdgeDetect+CAP1Mode*NegEdgeDetect
        .word  (_CAPINT0 + _CAPINT1) 0FFFFh  ;CONSTRUCT CAPTURE MASK
*
*
        .ref   IFimage              ;INTERRUPT FLAG IMAGE
        .ref   IMimage              ;INTERRUPT MASK IMAGE
        .ref   BANK                 ;TEMPORARY STORAGE FOR VALUE FOR BSR
        .ref   TMP                  ;SCRATCH SPACE
        .ref   POINT                ;TEMPORARY STORAGE FOR TABLE POINTER
        .ref   ONE                  ;THE VALUE 1 SAVED AS A CONSTANT
        .ref   SampleRate           ;NUMBER OF CLOCK TICKS FOR 0.5 MS
        .ref   MeanATime            ;MEAN CLK TICKS BETWEEN TRANSITIONS ON A
        .ref   MeanBTime            ;MEAN CLK TICKS BETWEEN TRANSITIONS ON B
        .ref   FIFO0value           ;LAST FIFO0 VALUE
        .ref   FIFO1value           ;LAST FIFO1 VALUE
        .ref   ABdifference         ;FIFO0 -FIFO1
        .ref   IntBankSave          ;PRESUMED INITIALIZED TO InterruptBank
*
        .bss   CapBankSave,1        ;DEFINE CAPTURE BANK
        .bss   TimBankSave,1        ;DEFINE TIMER BANK
*
* THE FOLLOWING CODE INITIALIZES CCON. IT WOULD BE REFERENCED DURING
* INITIALIZATION. USE TIMER 1 WITH INTERNAL CLOCK SOURCE WITH X1 PRE-
* SCALE. SET TIMER PERIOD TO 3125 (ABOUT 0.5 MS FOR A 25 MHZDEVICE).
*
        .text
*
CaptureInit:
*
        LT     ONE
        MPYK   TABLE
        PAC
        SACL   POINT
        TBLR   TMP
        LACK   CaptureBank
        SACL   CapBankSave          ;CapBankSave = CaptureBank
        OUT    CapBankSave,BSR
        OUT    TMP,CCON             ;INITIALIZE CCON FROM TABLE
*
        LAC    POINT
        ADD    ONE
        SACL   POINT
        LACK   TimerBank
        SACL   TimBankSave          ;TimBankSave = TimerBank
        OUT    TimBankSave,BSR
        OUT    TMP,TPR1             ;INITIALIZE TIMER 1 PERIOD
*
        LAC    POINT
        ADD    ONE
        SACL   POINT                ;POINT = ->TABLE[1]
        TBLR   TMP                  ;TMP = TABLE[1]
        OUT    IntBankSave,BSR
        IN     IMimage,IM           ;READ IM AND STORE IN IMimage
        LAC    TMP
```

```
       AND    IMimage
       SACL   IMimage
       OUT    IMimage,IM          ;ENABLE CAPINT0 AND CAPINT1
       RET
*
* CAPTURE ISR PRESUMES THAT A CAPTURE INTERRUPT WAS DETECTED
* AND THAT THE ENVIRONMENT HAS BEEN SAVED PRIOR TO THE DISPATCH
* OF THIS ROUTINE. IT IS PRESUMED THAT THE IF REGISTER WAS SAVED AT
* IFimage PRIOR TO IMAGE ENTRY INTO THE ROUTINE. UPON COMPLETION OF
* HANDLING THE CAPTURE INTERRUPT, THE ROUTINE EXECUTES A 'RET'
* INSTRUCTION, RETURNING IT TO THE POINT WHERE IT WAS INVOKED FOR THE
* RESTORATION OF THE ENVIRONMENT AND/OR FURTHER INTERRUPT PROCESSING.
*
CaptureISR:
       OUT    CapBankSave,BSR     ;BSR -> CAPTURE BANK
*
       LAC    ONE,CAPINT0_BIT
       AND    IFimage
       BNZ    Capture0            ;IF (CAPINT0) THEN PROCESS IT
       LAC    ONE,CAPINT1_BIT
       AND    IFimage
       BNZ    Capture1            ;ELSE IF (CAPINT1) THEN PROCESS IT
       RET                        ;RETURN TO CALLING ROUTINE
*
Capture0:
       IN     TMP,FIFO0
       LAC    TMP
       SUB    FIFO0value
       ADD    MeanATime
       SACL   MeanATime
       LAC    TMP
       SACL   FIFO0value          ;FIFO0value = FIFO0
       LAC    MeanATime,15
       SACH   MeanATime           ;MeanATime = (MeanATime + FIFO0dif)/2
*
       LAC    FIFO0value
       SUB    FIFO1value
       SACL   ABdifference        ;ABdifference = FIFO0value - FIFO1value
       RET
*
Capture1:
       IN     TMP,FIFO1           ;Get FIFO 1 VALUE
       LAC    TMP
       SUB    FIFO1value          ;SUBTRACT PREVIOUS FIFO1
       ADD    MeanBTime
       SACL   MeanBTime
       LAC    TMP
       SACL   FIFO1value          ;UPDATE FIFO1value
       LAC    MeanBTime,15
       SACH   MeanBTime           ;MeanBTime = (MeanBTime + FIFO1dif)/2
*
       RET
*
       .end
```

5.9 Using the Serial Port (TMS320C14)

The TMS320C14 has a dedicated asynchronous mode receiver/transmitter (UART). An independent timer is used for baud rate generation, if required. Two interrupts (receiver buffer full or transmitter buffer empty) are generated by the serial port to initiate transceiving.

The asynchronous operation is full-duplex with internal baud rate generation, with a maximum transmission rate of CLKOUT/16 bps (400 Kbps @ 25.6 MHz). In this mode, the serial port is capable of detecting parity, framing, and frame overrun errors.

5.9.1 Asynchronous Configuration

Example 5–50 below sets up the serial port in the asynchronous mode with 7 data bits, even parity, and normal reception. A software polling technique is used to determine if the receiver or the transmitter requires servicing, and data is either read from the receive buffer or written to the transmit buffer, depending on the source of the interrupt.

Example 5–50. Configuring for Asynchronous Operation

```
*
* EXAMPLE PROGRAM SEGMENT TO INITIALIZE SERIAL PORT FOR ASYNCHRONOUS OPERATION AND
* TO RESPOND TO INTERRUPTS FROM TRANSMIT AND RECEIVE SECTIONS OF THE SUBSYSTEMS.
*
* INCLUDE C14 EQUATES
*
   .include "C14INC"     ;INCLUDE HEADER FILE

*
     .def     AsynchInit,AsynchISR
     .data
*
TABLE:
     .word    SBRG19200     ;CONSTANT FOR 19.2 KBAUD
     .word    AsynchMode+SevenDataBits+ParityEnable+EvenParity
     .word    (_RXINT + _TXINT) ^ 0FFFFh
                            ;CONSTRUCT IM MASK FOR SERIAL PORT
*
     .ref     IFimage       ;INTERRUPT FLAG IMAGE
     .ref     IMimage       ;INTERRUPT MASK IMAGE
     .ref     BANK          ;TEMPOARAY STORAGE FOR BSR VALUE
     .ref     TMP           ;SCRATCH SPACE
     .ref     POINT         ;TEMPORARY STORAGE FOR TABLE POINTER
     .ref     ONE           ;THE VALUE 1 SAVED AS A CONSTANT
*
     .ref     PortBankSave  ;INITIALIZED TO SerialPort BANK
                            ;TO SPEED EXECUTION DURING
                            ;INTERRUPT SERVICE ROUTINE
*
     .ref     IntBankSave   ;INITIALIZED TO InterruptBank
                            ;TO SPEED EXECUTION DURING
                            ;INTERRUPT SERVICE ROUTINE
```

```
    .text
*
* THE FOLLOWING CODE INITIALIZES SCON. IT WOULD BE REFERENCED
* DURING INITIALIZATION.
*
AsynchInit:
*
       LT     ONE
       MPYK   TABLE
       PAC
       SACL   POINT            ;POINT = ->TABLE[0]
       TBLR   TMP
       LACK   SerialPort
       SACL   PortBankSave     ;PortBankSave = SerialPort
       OUT    PortBankSave,BSR
       OUT    TMP,SBRG         ;SET SBRG FOR 19.2 KBAUD
*
       LAC    POINT
       ADD    ONE
       SACL   POINT            ;POINT = ->TABLE[1]
       TBLR   TMP              ;TMP = TABLE[1]
       OUT    TMP,SCON         ;INITIALIZE SCON FROM TABLE
*
       LAC    POINT
       ADD    ONE
       SACL   POINT            ;POINT = ->TABLE[2]
       TBLR   TMP              ;TMP = TABLE[2]
       LACK   InterruptBank
       SACL   IntBankSave
       OUT    IntBankSave,BSR  ;IntBankSAve = InterruptBank
       OUT    TMP,IM           ;ENABLE RXINT AND TXINT
*
       RET
*
* THIS ROUTINE IS PRESUMED TO BE INVOKED FROM AN INTERRUPT
* SERVICE DISPATCHER WHICH SAVES AND RESTORES ENVIRONMENT
*
AsynchISR:
       OUT    PortBankSave,BSR ;BSR -> SERIAL PORT
*
       LAC    ONE, RXINT_BIT
       AND    IFimage
       BZ     Txtest           ;IF (RXINT) THEN PROCESS IT
*
* PROCESS INTERRUPT FROM RECEIVER
*
*    .
*    .
*    .
*
Txtest:
       LAC    ONE,TXINT_BIT
       AND    IFimage
       BZ     AsynchExit       ;ELSE IF (TXINT) THEN PROCESS IT
*
* PROCESS INTERRUPT FROM TRANSMITTER
*
*      .
*      .
*      .
*
```

```
AsynchExit:
       OUT    PortBankSave,BSR
       ZAC
       LAC    ONE,RXINT_BIT
       ADD    ONE,TXINT_BIT
       SACL   TMP
       OUT    TMP,FCLR          ;CLEAR INTERRUPT FLAGS FOR TX AND RX
       RET                      ;RETURN TO CALLING ROUTINE
*
       END
```

Chapter 6

Hardware Applications

Information and examples on interfacing a TMS320C1x (first-generation TMS320) digital signal processor with external devices are presented in this chapter. The examples given can be easily adapted to fit a particular system requirement.

Note:

Throughout this book, 'C14 designates all devices with a 14 suffix (C14/E14/P14), 'C15 all devices with a 15 suffix (C15/E15/LC15/P15), and 'C17 all devices with a 17 suffix (C17/E17/LC17/P17), unless otherwise noted.

This chapter includes the following topics

6.1 Expansion Memory Interfacing

The TMS320C1x can be interfaced to a wide variety of memory and I/O devices. The TMS320C10 and TMS320C15 devices can be interfaced to up to 4K words of external program memory. Expansion of program memory is accomplished directly through the use of the $\overline{MEN}$ (memory enable) and $\overline{WE}$ (write enable) control lines, with memory accesses occurring in a single cycle.

6.1.1 Interfacing (TMS320C14)

The TMS320C14 can be interfaced with PROMs, EPROMs, and RAMs. The TMS320C14 has no provisions to insert wait states; all memory devices should therefore meet the full-speed access requirements of the device.

The TMS320C14 implements two kinds of memory spaces: program memory (4K words) and data memory (256 words). Data memory accesses are always from internal RAM, and no off-chip accesses are available. Program memory accesses can be configured from either on-chip ROM/EPROM or off-chip memory. $\overline{REN}$ (read enable) and $\overline{WE}$ (write enable) strobes are active, whether accessing on-chip or off-chip program memory.

Read and Write Cycles

For timing diagrams, refer to the TMS320C14 data sheet. Memory interfaces assume that the TMS320C14 is running at 25.6 MHz and that Q is 39.1 ns. Q indicates one quarter cycle of CLKOUT frequency.

In the read cycle, the following sequence occurs (refer to Figure 6–1):

1) $\overline{REN}$ (or $\overline{WE}$) rises, terminating the previous bus cycle and starting the next cycle.
2) Address becomes valid no less than $t_{su(A)R}$ before $\overline{REN}$ goes low, to indicate the start of a read cycle.
3) The external device is selected with a minimum address setup time of $t_{a(A)}$ and puts its data on the bus.
4) Data must be valid for at least $t_{su(D)R}$ before $\overline{REN}$ rises and must be held valid for at least $t_{h(D)R}$.
5) $\overline{REN}$ goes high and the address bus becomes invalid after a delay of no more than $t_{h(A)}$.

For read-only devices, the maximum access time from address valid until data valid is:

$t_{a(A)} = tc_{(C)} - 90 = 156.25 - 90 = 66.25$ ns

From chip enable until data valid is:

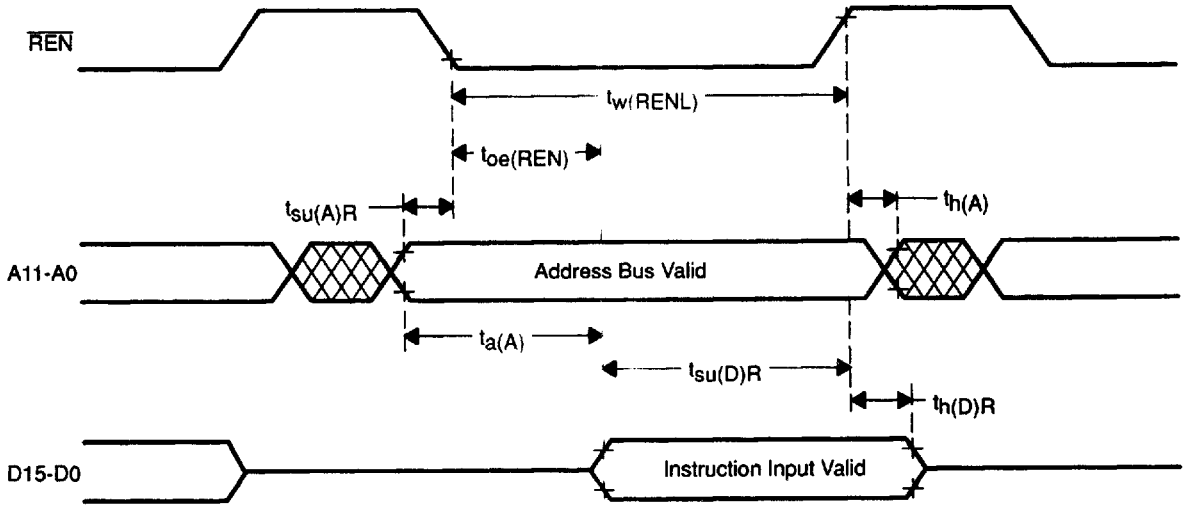
$t_{oe(REN)} = 0.75\, t_{c(C)} - 60$
$= 0.75(156.25) - 60 = 57.19$ ns

The minimum address setup time prior to chip enable will be:

$t_{su(A)R} = 0.25t_{c(C)} - 39 = 0.6$ ns

Therefore for read access, memories with a maximum access time of 57 ns from chip enable are required

Figure 6–1. Memory Read Timing



When a TBLW is executed, a write cycle is initiated and the following sequence occurs (refer to Figure 6–2):

1) $\overline{REN}$ (or $\overline{WE}$) rises, terminating the previous cycle and starting the write cycle for the TBLW instruction.
2) Address becomes valid after a minimum of $t_{su(A)W}$ before $\overline{WE}$ falling.
3) The data bus is driven after a maximum of $t_{en(D)W}$ before $\overline{WE}$ falls and data becomes valid after a delay of no more than $t_{su(D)W}$ before $\overline{WE}$ falls.
4) $\overline{WE}$ goes high and the address bus becomes invalid after a delay of $t_{h(D)W}$ after $\overline{WE}$ rising.
5) The data bus is driven to the high-impedance state after a delay of no more than $t_{dis(D)W}$ after $\overline{WE}$ rising.

For RAM devices, the maximum access time from address valid until data valid is:

$$t_{su(A)W} - t_{su(D)W} = 0.25\,t_{c(C)} - 45 - (0.25t_{c(C)} - 45)$$
$$= 0.25\,t_{c(C)} = 39.1 \text{ ns}$$

From chip enable until data valid is:

$t_{su(D)W} = 0.25\,t_{c(C)} - 45 = -5.9$ ns

The write enable access time is:

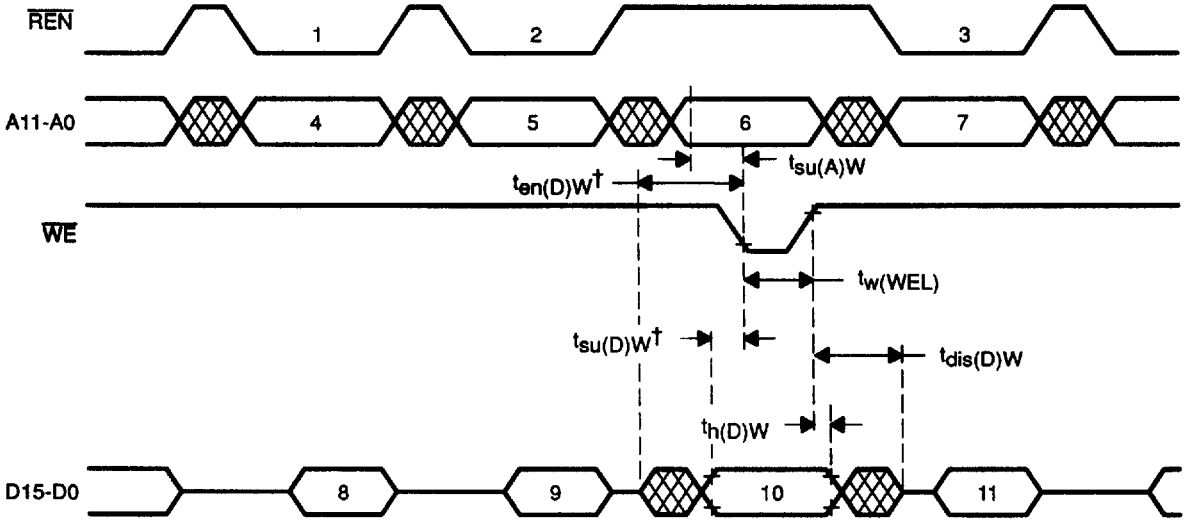
$t_{w(WEL)} = 0.5\ t_{c(C)} - 15 = 63.1$ ns

The minimum data hold time provided is:

$t_{h(D)W} = 0.25\ t_{c(C)} - 10 = 29.1$ ns

Therefore, for write access, memories with a maximum access time of 50 ns from chip enable are required.

Figure 6–2. Memory Write Timing (TBLW Instruction)



† Data valid prior to WE↓

LEGEND:

1. TBLW Instruction Prefetch
2. Dummy Prefetch
3. Next Instruction Prefetch
4. Address Bus Valid
5. Address Bus Valid
6. Address Bus Valid
7. Address Bus Valid
8. Instruction Input Valid
9. Instruction Input Valid
10. Data Output Valid
11. Instruction Input Valid

6.1.2 Program ROM Expansion

The address pins on a TMS329C1x device contain either the buffered outputs of the program counter or the I/O port address and are always driven. The bus control lines define the direction of data flow and the data space where the data belongs.

Read operations are performed on external memory either during opcode or operand fetches or during the execution of a TBLR (table read) instruction. Write operations have no effect on the circuit. When a read operation occurs on a TMS320C1x device, an address is placed on the address bus, and the

$\overline{MEN}$ (memory enable) strobe is generated by driving $\overline{MEN}$ low to enable external memory. The instruction word is then transferred to the TMS320C1x via the 16-bit data bus.

When a read operation occurs on the TMS320C14, an address is placed on the address bus, and the $\overline{REN}$ (read enable) strobe is generated by driving $\overline{REN}$ low to enable external memory. The instruction word is then transferred to the TMS320C14 via the 16-bit data bus.

Note:

The timing parameters of the 'C14, such as $t_{a(A)}$, are described in the data sheet in Appendix A. The use of CLOCKOUT as a reference is also eliminated in many cases. Note that the memory control signal names and functions are different from other 'C1x devices.

A memory address being placed on the bus becomes valid following a maximum delay (t_{d1}) from the falling edge of CLKOUT. The combined delay of

$t_{d1} + t_{a(A)} + t_{su(D)}$ = minimum cycle time, $t_{c(C)}$

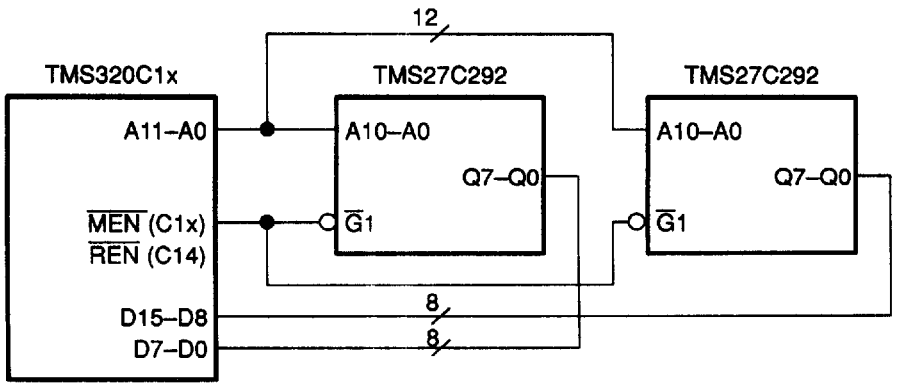
where:

$t_{a(A}$ = memory access time of EPROM from address valid
$t_{su(D)}$ = setup time form data bus valid prior to CLKOUT↓.

serves as the timing constraint used when calculating $t_{c(C)}$.

When only external program ROM is required, a minimum system can consist of a TMS320C1x with 4K words of external program memory (TMS27C292), as shown in Figure 6–3. The $\overline{MEN}$ ($\overline{REN}$ on the 'C14) signal and the address (A11–A0) and data (D15–D0) lines on the TMS320C1x are connected directly to the TMS27C292 memories, and no address decoding is required. These memories are a pair of TMS27C292 4K × 8 ROMs by Texas Instruments, configured in parallel for a direct 16-bit interface to the TMS320C1x. The TMS320C16 is not shown but it can access up to 64K of external memory. In this case address-decoding logic may be needed.

Figure 6–3. Minimum Program ROM Expansion

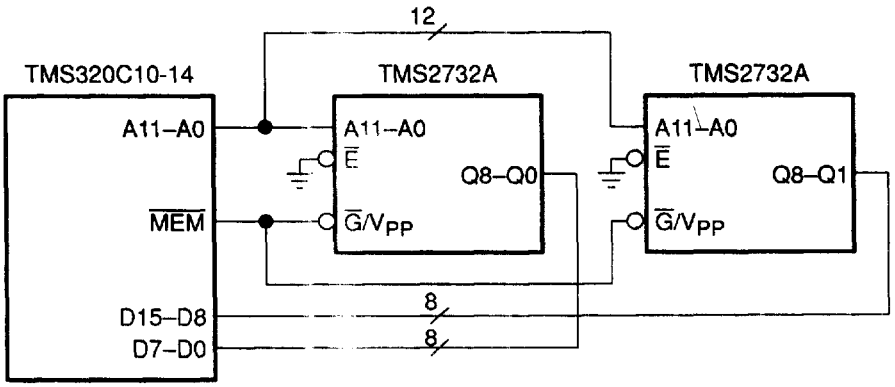


An inexpensive system with minimal chip-count is possible with the TMS320C10-14. Using an EPROM interfaced to a TMS320C10-14 in external program memory allows the implementation of 4K words of nonvolatile program memory and gives added flexibility in reprogramming. This in turn, permits system development, future program expansion, and/or upgrade modification. Single-cycle memory access with a direct memory interface requires no additional external interface logic.

On the TMS320C10-14, t_{d1} with a maximum value of 50 ns and $t_{su(D)}$ with a minimum value of 50 ns are both constants; therefore, $t_{a(A)}$ is the only remaining variable used in determining the minimum clock cycle time of the system. For the circuit shown in Figure 6–4 (with $t_{a(A)}$ = 170 ns), inserting these values into the equation yields $t_{c(C)}$ min = 270 ns.

In Figure 6–4, a pair of Texas Instruments TMS2732A-17 4K × 8 EPROM memories are configured in parallel for a direct 16-bit interfacing with TMS320C10-14. These EPROMs display a 170-ns access time. However, other EPROMs may be used with access times best suited to a particular application, as long as the TMS320C10-14 clock frequency has been selected to allow for the access time of the EPROMs chosen.

Figure 6–4. EPROM Interface to the TMS320C10-14



Contention for the data bus is not a concern in this memory configuration. Therefore, the $\overline{E}$ (chip enable) pin for the EPROM pair has been tied to ground to avoid unnecessary switching transients that could be induced if the chip enables were toggled upon memory access.

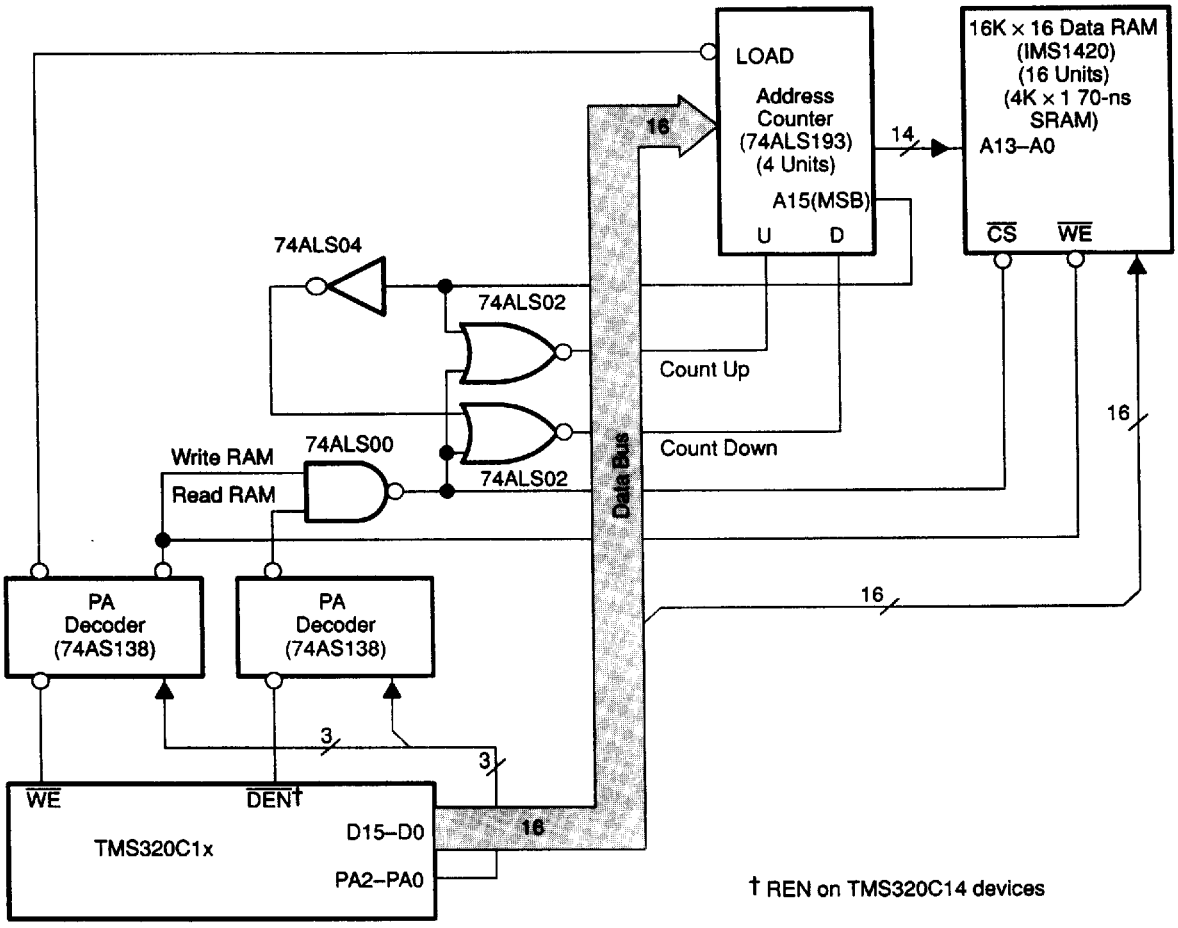
6.1.3 Data RAM Expansion

No direct memory expansion is provided on the TMS320C1x. However, if RAM is used for external program memory, this memory can be used to store data information accessed with the TBLR and TBLW instructions. These instructions, however, take three cycles to execute.

If larger memory or faster memory accesses are required, an alternative memory expansion scheme using I/O ports can be implemented for a TMS320C1x device. In this case, additional RAM can supplement internal data memory and can be accessed in only two cycles through the IN and OUT instructions. If RAM is to be used for program memory, additional logic must be included to distinguish between an I/O write (OUT) and a program memory write (TBLW).

Figure 6–4 provides an example of external data memory expansion. The design consists of up to 16K words of static RAM (IMS1420), addressed by the lower 14 bits of a 16-bit counter (74ALS193). In the case of the IMS1420s, the address of the data to be accessed is loaded into the counter by implementing an OUT instruction to port 0. This loads the data bus into the counters. Memory can then be read from or written to sequentially by performing an IN or OUT instruction to port 1. The MSB in the counters determines whether the memory address is increased (MSB = 0) or decreased (MSB = 1) after a read or write of data memory. Memory continues to be addressed sequentially until new data is loaded into the counters.

Figure 6–5. Data RAM Expansion



Dynamic memories may also be used for external data memory expansion; however, these devices may impose additional constraints on the system design. For example, some memory cycle times may not allow consecutive IN/OUT/IN instruction sequences. Memory refresh must also be considered. Because the TMS320C1x does not implement wait states, memory refresh must be generated transparently to the processor.

For additional information regarding interfacing to TMS320C1x devices, refer to *Digital Signal Processing Applications with the TMS320 Family, Volume 1* (literature number SPRA012A).

6.2 External Peripheral Interfacing

The TMS320C1x is flexible enough to be used in a wide range of applications requiring different types of peripheral interfaces. The following subsections describe A/D and D/A interfaces, codec interfaces and a serial communication RS-232 interface.

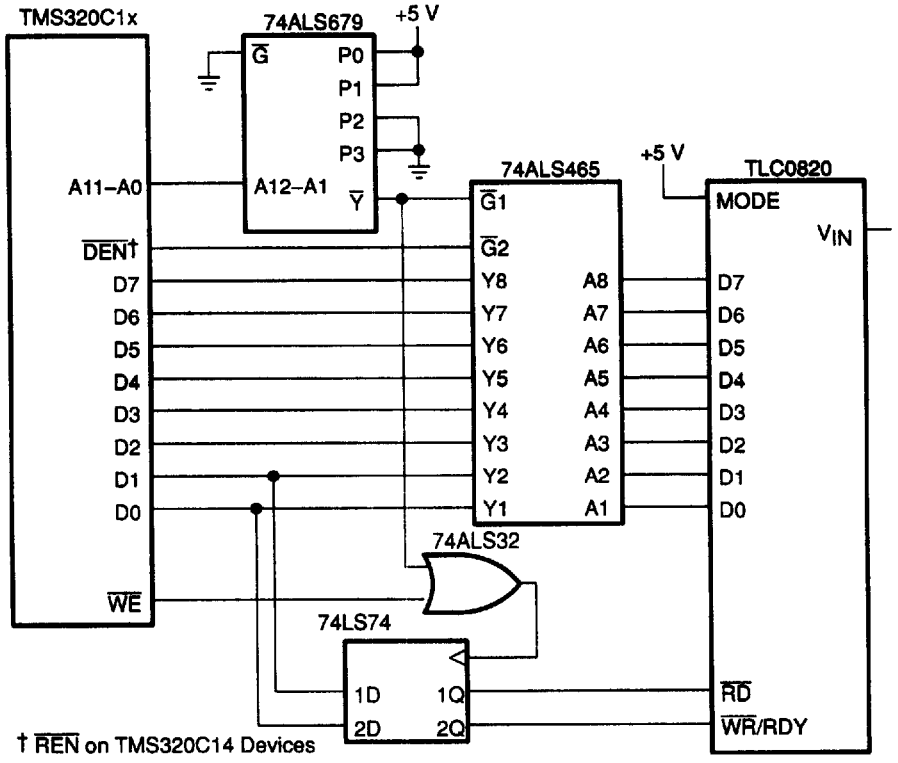
6.2.1 A/D Interface

The TMS320C1x can be interfaced to an A/D (analog-to-digital) converter to perform the necessary conversions. Normally, only a minimum of external circuitry is required.

Figure 6–6 shows an interface of the TLC0820 8-bit A/D converter to the TMS320C1x. Because the control circuitry of the TLC0820 operates much more slowly than the TMS320C1x, it cannot be directly interfaced. All of the logic functions are implemented with one each of the following devices from the 74ALS family of advanced low-power Schottky logic:

74ALS679	12-bit address comparator
74LS74	Dual positive edge-triggered D-type flip flops
74ALS465	Octal buffer with three-state output
74ALS32	Quad two-input OR gate

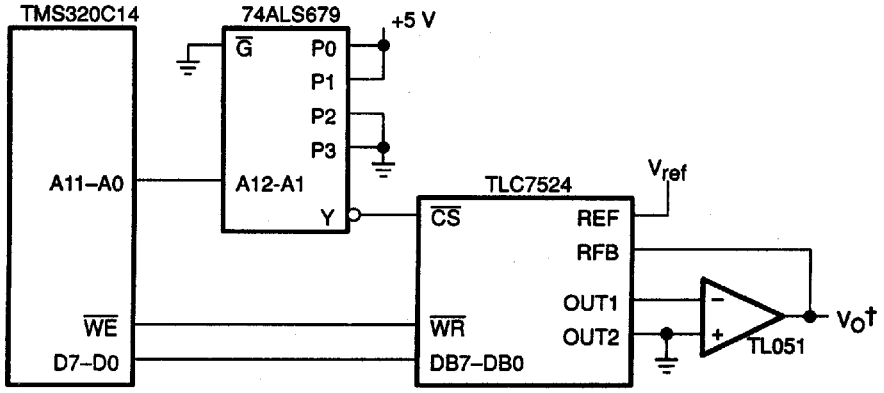
Figure 6–6. A/D Converter to TMS320C14 Interface



6.2.2 D/A Interface

A memory-mapped interface of the TLC7524 8-bit D/A converter to the 'C14 is shown in Figure 6–7. Because table writes and I/O write cycles use the same external control signals, A12-bit address decoder is used to avoid memory and I/O conflicts. In the configuration shown in Figure 6–7, I/O port address zero is configured to a corresponding address of 03F8h.

Figure 6–7. D/A Converter to TMS320C14 Interface



† $V_O = -V_{ref} \frac{D}{256}$, where D = digital input

For further information about the A/D and D/A converters shown in the figures, refer to the *Linear Circuits Data Book* (literature number SLYD001).

6.2.3 Codec Interface (TMS320C17)

Some areas of speech, telecommunications, and other applications require low-cost analog-to-digital (A/D) and digital-to-analog (D/A) converters. A combo-codec is a nonlinear single-chip PCM (pulse-code modulated) codec with antialiasing and smoothing filters and data storage registers. It encodes (A/D) and decodes (D/A) efficiently and economically in a single 300-mil DIP package. For additional information on combo-codecs, refer to the *TCM29C13/C14/C16/C17 Combined Single-Chip PCM Codec and Filter* data sheet.

The TMS320C17 is capable of direct interface to serial devices such as combo-codecs, thus reducing chip count and improving system throughput. By using on-chip hardware, the C17 can also compand (COMpress and exPAND) a PCM data stream acquired by the codec.

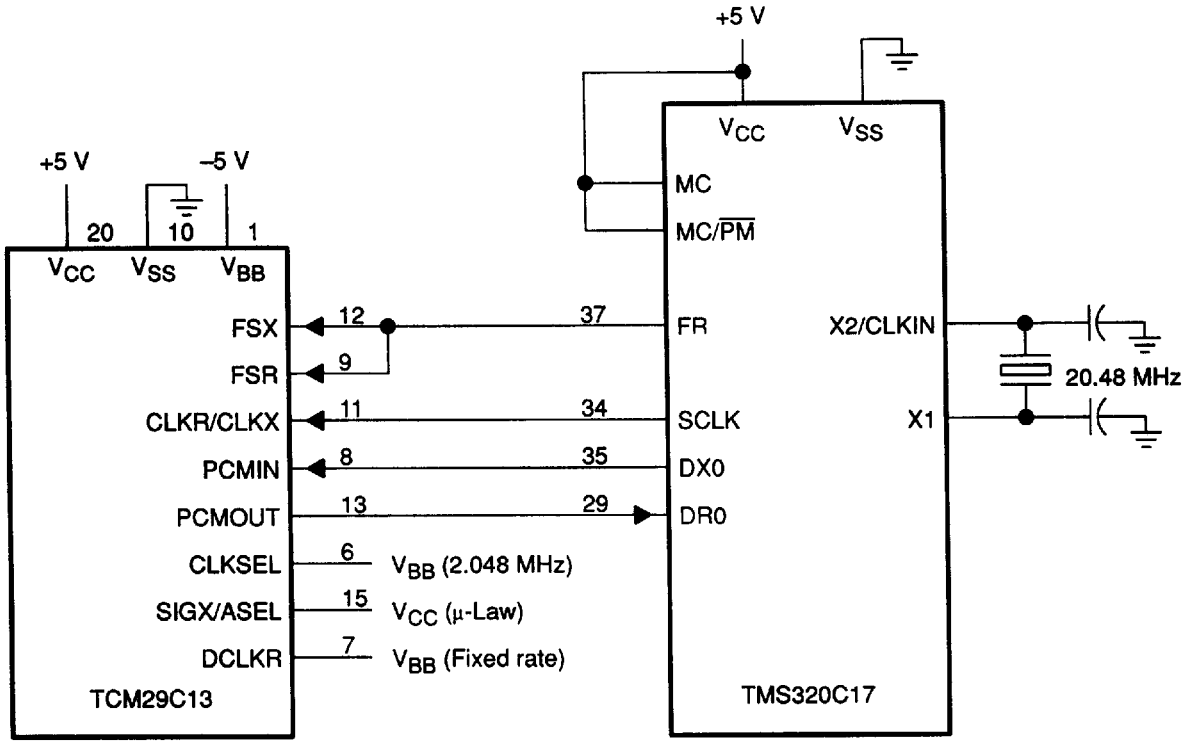
Figure 6–8 shows the TMS320C17 in a standalone full-duplex direct serial port interface with a TCM29C13 combo-codec. The TMS320C17 provides the serial clock for bit transmission. The codec is sampled every 125 µs (8-kHz frequency), at which time an 8-bit PCM byte is exchanged between the two devices. A second codec can also be interfaced to the TMS320C17 with no additional logic or interconnections because these devices implement two independent serial ports.

Timing for the serial interface system is controlled by the serial port clock (SCLK). SCLK is configured as an output from the TMS320C17, and its fre-

quency is set to 2.048 MHz (see Section 3.13). A 20.48-MHz crystal is input to the TMS320 as its system clock. The SCLK frequency is derived from this system clock by a divide-by-10 in the SCLK prescale control logic, initialized through control register 1. SCLK is connected to CLKR/CLKX on the TCM29C13 to provide the transmit-and-receive master clock. CLKSEL on the codec is tied to V_{BB} to select the 2.048-MHz master clock mode.

Framing pulses are generated by the TMS320C17 on the FR output pin. The frequency of these pulses is set to 8 kHz by dividing the serial clock (SCLK) by 256. This value is also initialized through control register 1. The short FR framing pulses supply the codec with framing pulses for the fixed data-rate mode. FR is input to both the FSX and FSR inputs on the codec. The FR output causes simultaneous transmit and receive operations from the serial port. The FSX input on the codec causes the device to transmit PCM data on the next eight consecutive positive transitions of the serial port clock (SCLK). The FSR input on the codec causes the device to receive PCM data on the next eight consecutive negative transitions of the serial port clock (SCLK). With this timing, the codec transmits and receives one 8-bit PCM sample every 125 μs.

Figure 6–8. Codec Interface for Standalone Serial Operation



The TMS320C17 transmits its PCM sample via the DX0 pin and is received by the TCM29C13 on the PCM IN pin. The TMS320C17 receives PCM sam-

ples on its DR0 pin, from the PCM OUT pin of the TCM29C13. This is a single-channel operation. All data transmission occurs on channel 0, requiring one IN instruction from port 1 to receive the PCM sample and one OUT instruction to port 1 to send a sample to the codec.

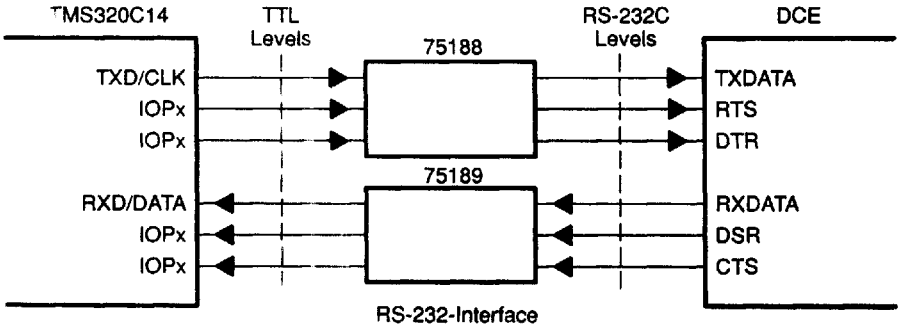
In the serial interface configuration, μ-law companding is selected by setting system control register bit 14 (CR14) to logic 0. The TCM29C13 is put into the μ-law companding mode by connecting the SIGX/ASEL pin to V_{CC}.

Linear A/D and D/A converters may also be interfaced to the TMS320C17 through its parallel ports instead of through the serial port.

6.2.4 RS-232 Interface

The TMS320C14 supports implementation of an RS-232 interface for connection to communication equipment, terminals, and PCs. The only devices needed are line drivers/receivers for the TTL/RS-232 level conversions. Figure 6–9 shows a typical interface with the TMS320C14 as a data terminal equipment (DTE) device connected to a data communication equipment (DCE) device (such as a modem). The serial port is configured for the asynchronous mode with the appropriate parameters selected. The bit I/O pins IOP15-IOP0 provide any necessary handshaking signals (that is, RTS, CTS, DSR, and DTR).

Figure 6–9. RS-232 Interface

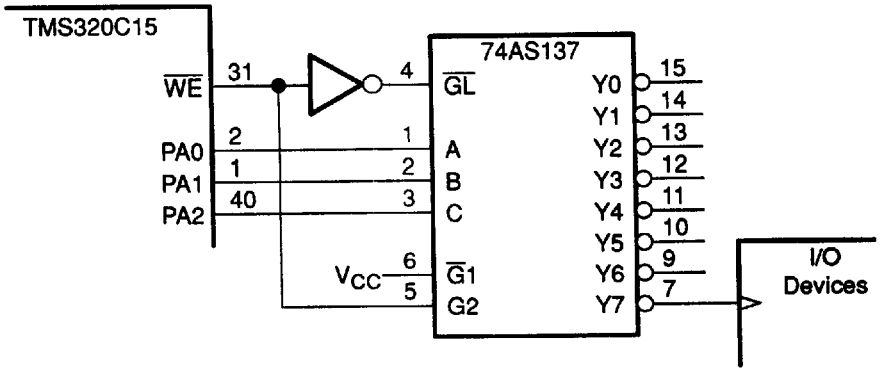


6.3 I/O Ports

The TMS320C1x devices interface to input/output (I/O) devices through the eight 16-bit parallel ports (see Section 3.6 for I/O functions). The I/O space is selected by the appropriate control signals, such as $\overline{DEN}$ for reads and $\overline{WE}$ for writes on the TMS320C15. Each of the eight I/O ports is addressed by the three LSBs of the address bus with all other address lines held low, or high for the TMS320C14. The I/O ports share the 16 data lines.

The I/O ports may be used for interfacing external circuitry such as data memory expansion devices (see Section 6.1), A/D and D/A converters, synchronization latches, or memory-mapped peripheral devices. Figure 6–10 shows a circuit that can be used to generate device select lines for each of the individual port writes. A similar circuit may be used to enable I/O port reads.

Figure 6–10. I/O Port Interface Circuit



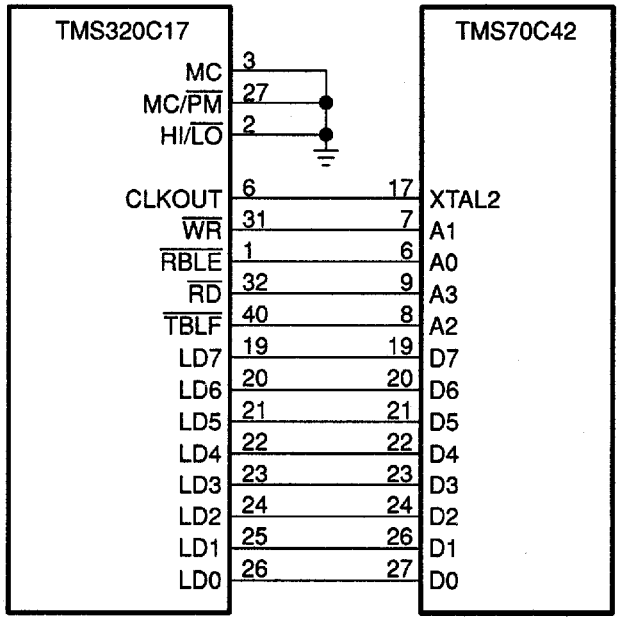
When you interface the TMS320C1x to slower devices, a handshake interface used in conjunction with the I/O port interface may be desirable. Data to be transferred may be stored in latches to be read by the TMS320C1x at a later time. Handshaking may then be established with the interrupt, $\overline{BIO}$, and XF (TMS320C17) signals.

6.4 Coprocessor Interface (TMS320C17)

The TMS320C17 offers the option to use the parallel I/O interface exclusively as a coprocessor interface. This option includes both the buffer logic to communicate between two processors asynchronously, and the protocol logic to protect against poor communication. The coprocessor port allows the TMS320C17 to act as either a master processor or a slave processor in a multiprocessing system. The circuit also allows data to be transferred as either 8- or 16-bit values.

As a master processor, the TMS320C17 writes to and reads from the coprocessor interface at will. This requires that the slave processor keep the receive buffer full and the transmit buffer empty. Figure 6–11 shows the TMS320C17 as a master processor to a TMS70C42 (8-bit microcomputer). As the internal CPU writes to the coprocessor interface, the $\overline{\text{TBLF}}$ (transmit buffer latch full) signal is driven active low. This signals the TMS70C42 that there is data to be read and that the 8-bit microcomputer must read that data before the next write by the internal CPU. In Figure 6–11, the $\overline{\text{TBLF}}$ signal is tied to an I/O bit on the 8-bit microcomputer so that the microcomputer can poll the signal and act accordingly. This signal can also be tied to an interrupt on the 8-bit microcomputer if this suits system requirements better. When the internal CPU reads its buffer, it signals the 8-bit microcomputer that the read buffer is empty by generating the $\overline{\text{RBLE}}$ (read buffer latch empty) signal. This signals the microcomputer that it must reload the receive latch before the next internal CPU access.

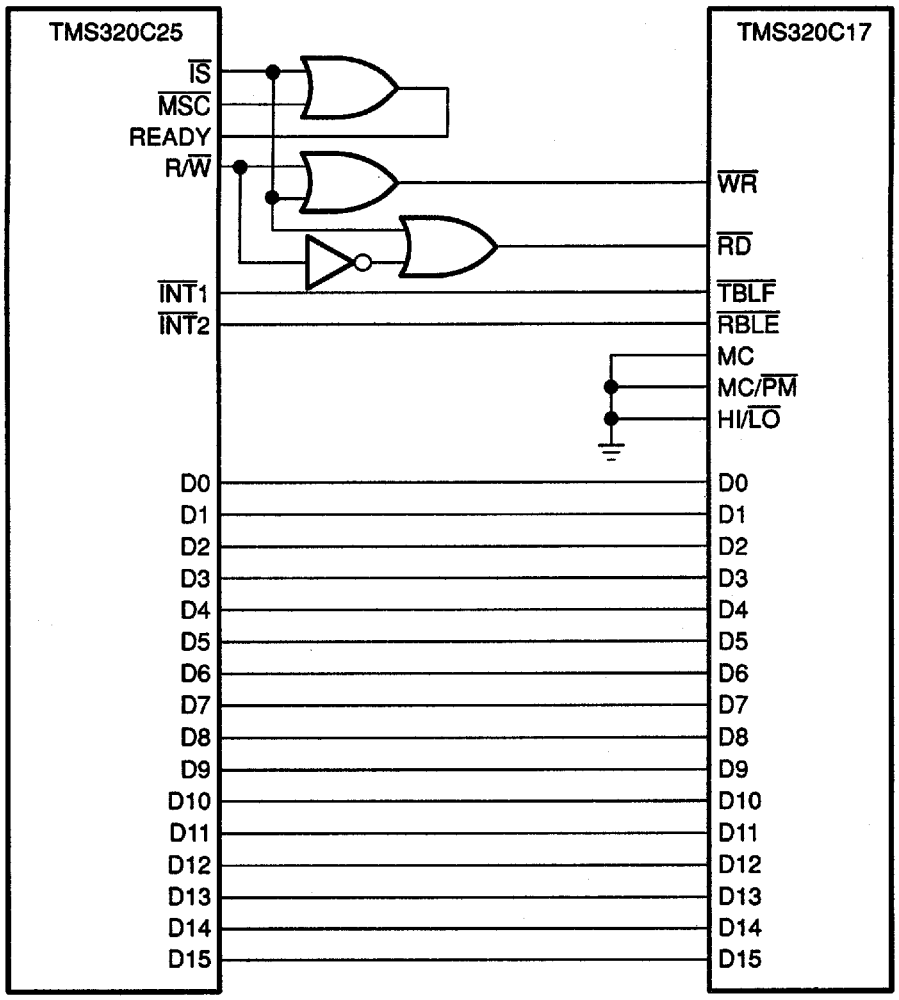
Figure 6–11. TMS320C17 to TMS70C42 Interface



When the TMS320C17 serves as a slave processor, transfer of data is controlled by a master processor. Figure 6–12 shows how the TMS320C17 (slave) interfaces with the 16-bit microprocessor TMS320C25 (master). When TMS320C25 writes to the TMS320C17, an interrupt signal is sent from the master to the internal CPU of the slave. The CPU must then read the information stored in the coprocessor interface before the next write from the TMS320C25. When the TMS320C25 reads the transfer latch of the coprocessor port, the internal CPU of the slave receives an active low $\overline{BIO}$ signal. When information is transferred to the master processor, the internal CPU monitors the $\overline{BIO}$ line (using the BIOZ instruction) to determine when it can reload the transmit latch. Note that a wait state may be required when interfacing to the TMS320C25.

To support mixed 8-/16-bit operation, the read buffer latch is cleared to 0 when read by the internal CPU.

Figure 6–12. TMS320C17 to TMS320C25 Interface



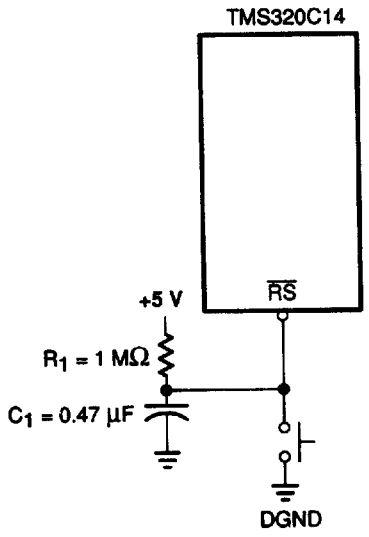
6.5 System Control Circuitry (TMS320C14)

The system control circuitry performs functions that are critical for proper system initialization and operation. A powerup reset circuit design and a crystal oscillator circuit design are presented in this section. The powerup reset circuit assures that a reset of the part occurs only after the oscillator is running and stabilized.

6.5.1 Power-Up Reset Circuit

The reset circuit shown in Figure 6–13 performs a powerup reset; that is, the TMS320C14 is reset when power is applied. Note that the switch circuit may include debounce circuitry. Driving the $\overline{RS}$ signal low initializes the processor. Reset affects several registers and status bits. (refer to subsection 3.5.2 for a detailed description of the effect of a reset on processor status).

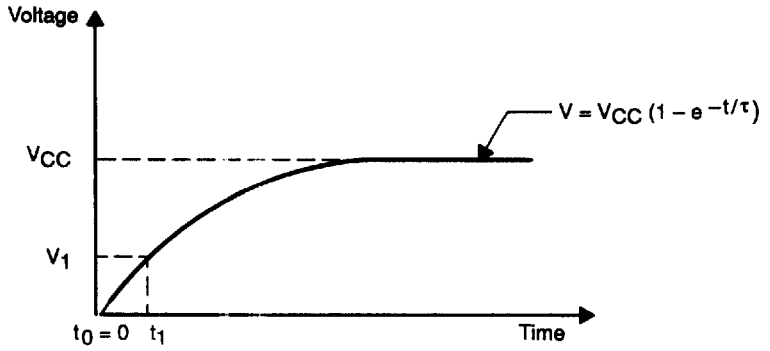
Figure 6–13. Power-Up Reset Circuit



For proper system initialization, the reset signal must be applied for at least five CLKOUT cycles, that is, 800 ns for a TMS320C14 operating at 25.6 MHz. Upon powerup, it can take up to one hundred milliseconds before the system oscillator reaches a stable operating state. Therefore, the powerup reset circuit should generate a low pulse on the reset line until the oscillator is stable (that is, 100 to 200 ms).

The voltage on the reset pin ($\overline{RS}$) is controlled by the R_1C_1 network (see Figure 6–13). After a reset, this voltage rises exponentially according to the time constant R_1C_1, as shown in Figure 6–14.

Figure 6–14. Voltage on TMS320C14 Reset Pin



The duration of the low pulse on the reset pin is approximately t_1, which is the time it takes for capacitor C_1 to be charged to 1.5 volts. This is approximately the voltage at which the rest input switches from a logic level 0 to a logic level 1. The capacitor voltage is given by following formula:

$$V = V_{CC}\left[1 - e^{-t/\tau}\right]$$

Where $\tau = R_1C_1$ is the reset circuit time constant.

Solving the equation for t is given in the formula:

$$t = -R_1C_1 \ln\left[1 - \frac{V}{V_{CC}}\right]$$

For example, the following values of:

$R_1 = 1\ M\Omega$ $\quad V_{CC} = 5\ V$
$C_1 = 0.47\ \mu F$ $\quad V = V_1 = 1.5\ V$

yield $t = t_1 = 167$ ms. In this case, the reset circuit of Figure 6–13 can generate a low pulse of long enough duration (167 ms) to ensure the stabilization of the oscillator upon powerup, in most systems.

6.5.2 Crystal Oscillator Circuit

The TMS320C14 requires an external clock source to drive the CLKIN pin. Either a prepackaged oscillator can be used or a simple oscillator circuit with TTL gates can be built. Prepackaged oscillators provide a wide operating range and better stability. A well-designed crystal oscillator, however, will provide good performance with TTL gates. Two types of crystal oscillator circuits can be used; one with series resonance, or one with parallel resonance.

Figure 6–15 shows implementation of a parallel resonant oscillator circuit. The circuit is designed to use the fundamental frequency of the crystal. The 74AS04 inverter performs the 180-degree phase shift that a parallel oscillator requires. The 4.7-kΩ resistor provides the negative feedback for stability, that is, the poles of the system are constrained in a narrow region about the Jω axis of the s-plane (analog domain). The 10 KΩ potentiometer biases the 74AS04 in the linear region.

Figure 6–15. Parallel Resonant Crystal Oscillator Circuit

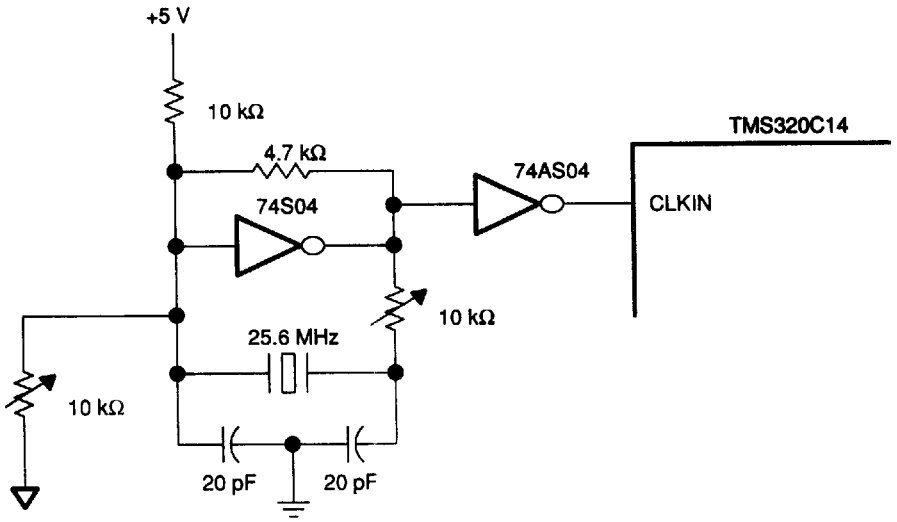
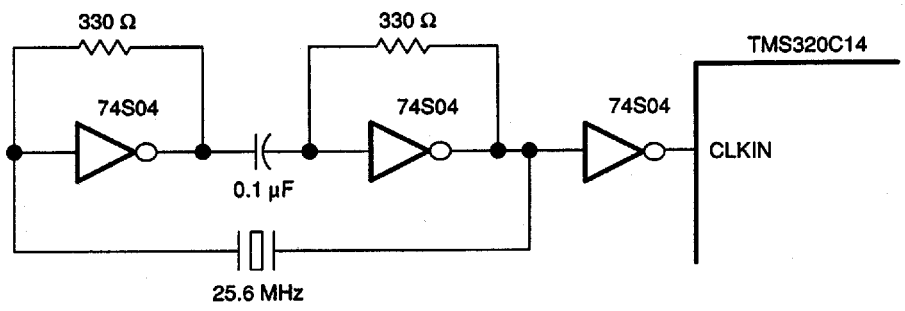


Figure 6–16 shows a series resonant oscillator circuit. This circuit is also designed to use the fundamental frequency of the crystal. The inverter performs a 180-degree phase shift in a series resonant oscillator circuit. The 330-Ω resistors provide the negative feedback to bias the inverters in their linear region.

Figure 6–16. Series Resonant Crystal Oscillator Circuit

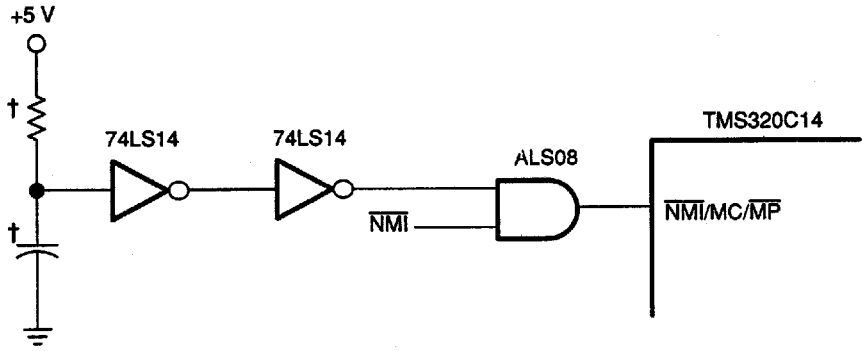


6.5.3 MC/$\overline{\text{MP}}$ Mode Configurations

The TMS320C14 can be configured (by software or hardware) to address internal or external program memory. The software technique uses the MC/$\overline{\text{MP}}$ bit in the SYSCON register to switch between internal and external memory. The hardware technique uses the $\overline{\text{NMI}}$/MC/$\overline{\text{MP}}$ pin, which is sensed during reset. If the pin is low, the device is put in the microprocessor (external memory) mode. If the pin is high, the device is put in the microcomputer (internal memory) mode.

Figure 6–17 shows a circuit for configuring the device into the microprocessor mode using a hardware technique. The $\overline{\text{RS}}$ and $\overline{\text{NMI}}$/MC/$\overline{\text{MP}}$ pins cannot be tied together since the $\overline{\text{NMI}}$/MC/$\overline{\text{MP}}$ pin must be held low for 1.25 clock cycles (200 µs in the circuit below) after $\overline{\text{RS}}$ goes high. If the $\overline{\text{NMI}}$ function is not being used, however, the $\overline{\text{NMI}}$/MC/$\overline{\text{MP}}$ pin may be tied to ground. This will not produce an interrupt because the $\overline{\text{NMI}}$ function is edge-triggered.

Figure 6–17. Mode Control Circuit



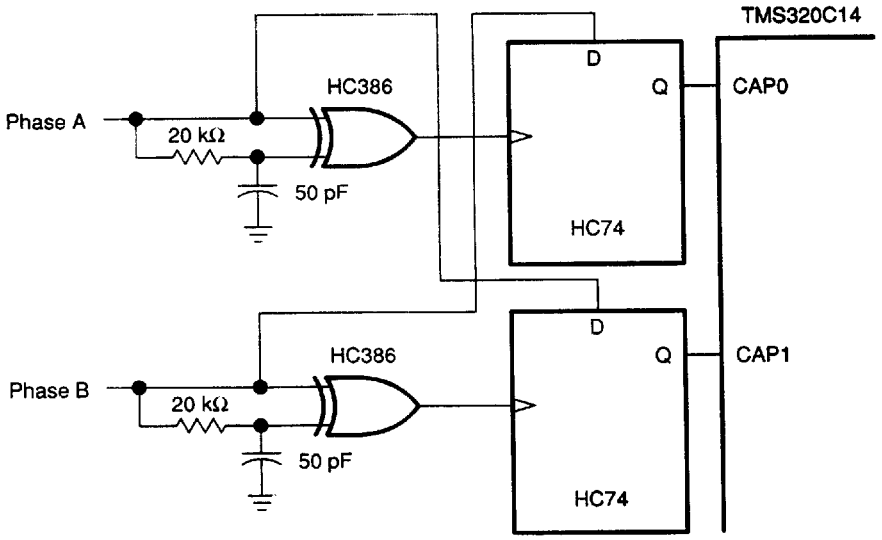
† Values depend on reset timing. To operate with this circuit, use values of 1 MΩ and 1.33 µF.

6.5.4 Optical Encoder Interface

Optical encoders are commonly used as sensors in measuring both speed and position. The capture inputs of the TMS320C14 interface directly with optical

encoders. A very cost-effective noise filtering system can be built with gates, D latches, and RC filters. The one-shot circuit constructed with two XOR gates generates pulses on each edge of each signal. Pulses generated by phase A are used to clock the signal from phase B, and vice versa. Figure 6–18 shows a typical interface with an optical encoder.

Figure 6–18. Optical Encoder Interface



6.5.5 XDS Design Considerations

The TMS320C14 XDS emulator is implemented with a dual processor system incorporating a TMS320C14 and a TMS9996. The TMS9996 acts as controller in the system, while the TMS320C14 performs the emulation. The design of the XDS maximizes performance and allows full-speed in-circuit emulation. This discussion covers general design considerations as well as timing and loading.

6.5.5.1 Bus Control

When the emulator is halted from the keyboard or by a breakpoint condition, the current state of the TMS320C14 is extracted by the TMS9996, implemented to look like a coprocessor. The TMS9996 communicates with the TMS320C14 over the internal data bus (of the emulated device), which is not seen by the user. Additional communication between the two processors is generated with commands entered from the keyboard. The TMS9996 shares the data bus only when the $\overline{REN}$ and $\overline{WE}$ signals are high. The target system should drive the data bus *only* when devices on the target system are addressed and $\overline{REN}$ or $\overline{WE}$ is low. If these rules are violated, the XDS gives a PROCESSOR SYNC LOST error message #1185. This error may also be

caused by signal-to-signal shorts in the target system, misalignment of the target connector, or wiring errors in the target system.

6.5.5.2 XDS Timing

The TMS320C14 emulator has additional logic in series with the inputs $\overline{RS}$, $\overline{INT}$, and $\overline{NMI}$/MC/$\overline{MP}$ and with outputs $\overline{REN}$ and $\overline{WE}$. Propagation delays assume 3 ns for cable delay and capacitive loading. The delays outlined in Table 6–1 are in addition to device specifications.

Table 6–1. XDS/Target Device Timing Delays

Signal	Delay
$\overline{INT}$, $\overline{NMI}$, $\overline{RS}$	Asynchronous
$\overline{WE}$, $\overline{REN}$, PA2–PA0, CLKOUT	Delay from part to pin = 7 ns
CLKIN	PLCC target cable delay from pin to part = 20 ns
Other Signals	Delay from part to pin or pin to part = 3 ns

6.5.5.3 Reset

$\overline{RS}$ is synchronized on the rising edge of CLKOUT after being sent from the pin through an array logic device. RS is applied to the TMS320C14 on the rising edge of CLKOUT. This is done to implement a RUN upon application of the target reset function in the emulator. It is not necessary for you to synchronize $\overline{RS}$.

6.5.5.4 Emulator Loading

Additional loading on the outputs is induced by the XDS. The differences in the DC loading between the emulator and the device are defined Table 6–1 below.

Table 6–2. XDS/ Device DC Loading

Signal	Load
$\overline{RS}$	0.3 mA
$\overline{NMI}$/MC/$\overline{MP}$	0.6 mA
D15–D0	1.0 mA
$\overline{REN}$, $\overline{WE}$	0.3 mA

6.5.5.5 Transmission Line Phenomena

Because the XDS target cable is approximately 20 inches long, use of advanced CMOS or fast/advanced Schottky TTL may cause line reflections (ringing above input thresholds) on input lines to the XDS. Series termination resis-

tors (22 to 68 ohms) can help eliminate this problem. Generally, you should use the cables as supplied, and keep connections as short as possible.

6.5.5.6 Clock Source

The XDS emulator uses only an internal oscillator or a TTL clock source provided by the target system. The emulator clock can be selected from three sources:

- The target clock
- A socketed, changeable crystal (Y2) on the processor module (PM) board
- A socketed, changeable canned TTL oscillator (U1) on the PM board

6.5.5.7 Miscellaneous Considerations

The XDS emulator initially sets up the emulated device to run and does not communicate with it until you activate the emulator via the keyboard. If the target system is continuously asserting $\overline{RS}$, the XDS does not gain control of the device and reports a PROCESSOR SYNC LOST error message #1185. This condition can be caused by a powered-up emulator plugged into a powered-down target system. Even though $\overline{RS}$ is pulled up through a resistor on the emulator, the impedance of the powered off target system can be low enough to assert a low on $\overline{RS}$ or load the data bus so as to keep the emulator from functioning.

> The conductive foam on the XDS target connector and logic cable must be removed before power-up. Failure to do so will result in a malfunction and may cause damage.

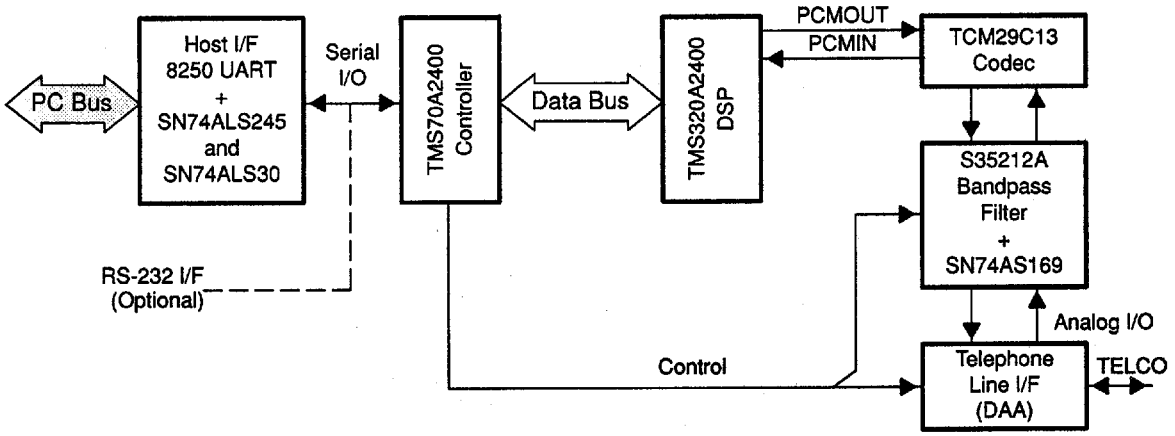
6.6 TMS320C1x System Applications

Several system applications are presented in this section in a general form to illustrate basic approaches to system design with the TMS320C1x. These applications include a 2400-bps modem, a speech synthesis system, and a voice store-and-forward message center.

6.6.1 2400-bps Modem

The implementation of a 2400-bps modem is shown in Figure 6–19. This system implements the functions of a V.22 bis modem using a TMS320A2400 and a TMS70A2400, which are masked ROM versions of the TMS320C17 and TMS7042, respectively. The TMS320A2400 performs all of the signal processing functions, and the TMS70A2400 performs all of the interface protocol and control functions. The remaining system components perform the necessary A/D and D/A conversions as well as the PC bus interfacing, telephone line interfacing, and filtering functions.

Figure 6–19. 2400-bps Modem



6.6.2 Speech Synthesis System

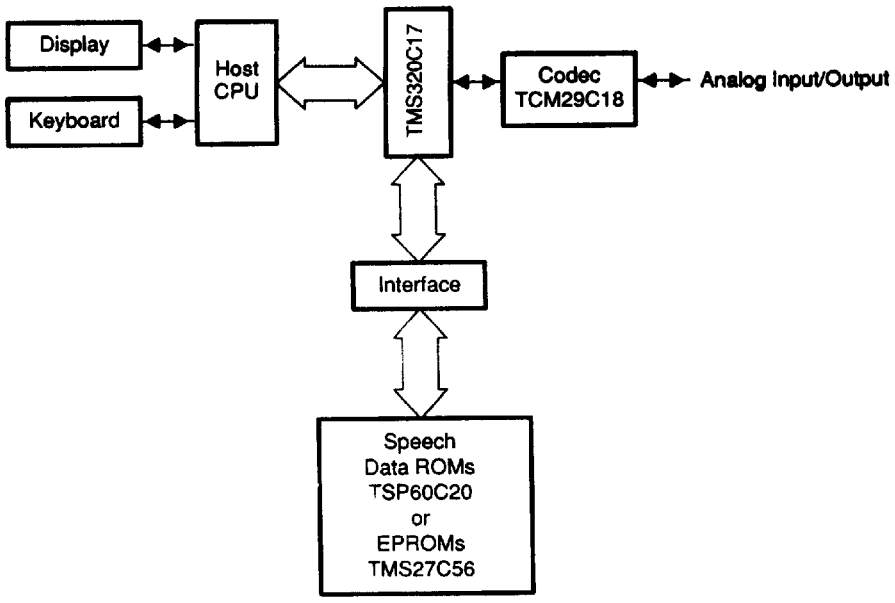
The system design for speech applications consists of a codec, a digital signal processor supported with program and data memory, a speech data memory, and an optional host processor. A block diagram of this system, shown in Figure 6–20, consists of the following components:

- Codec (TCM29C18)
- Digital signal processor (TMS320C17)
- Speech data ROM (TSP60C20) or EPROM (TMS27C56)
- Microcomputer host (TMS70C42)

The actual speech system is composed of the digital signal processor and the codec. The microcomputer host is performs an end-product application that

calls upon the speech subsystem when needed, such as in the case of a minicomputer and array processor system. The speech system can perform speech synthesis, vocoding, speech recognition, speaker verification, and DTMF decoding/encoding as well as many other algorithmically intensive applications.

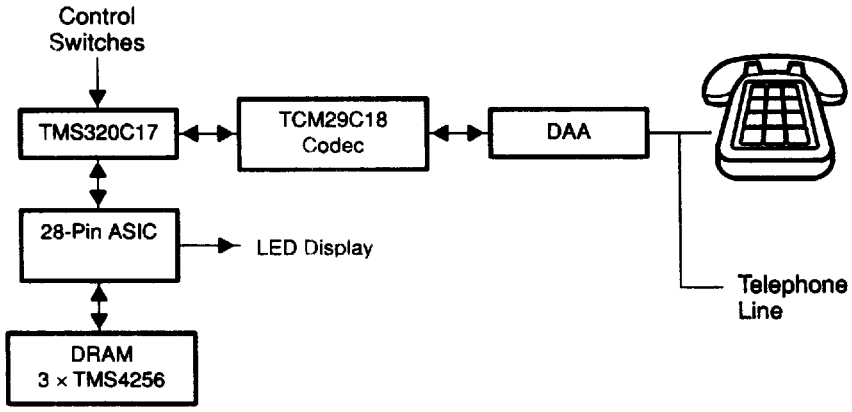
Figure 6–20. Speech Synthesis System



6.6.3 Voice Store-and-Forward Message Center

The voice store-and-forward message center consists of a TMS320C17-based system interfaced to a phone line and a large storage area either on DRAMs or computer disks, depending on the application. Some applications of the message center are voice mail for a computer network, answering machines for home use (see Figure 6–21), and a hand-held battery-operated voice message pad for personal use. Typical algorithms required to perform the task are half-duplex ADPCM or sub-band coder, LPC synthesis, and DTMF encoder/decoder. A combination of these algorithms fits into the 4K on-chip program ROM of the TMS320C17 and requires no external data memory. Because the CPU utilization is less than 100 percent when performing any of these tasks, other operations can also be done by the TMS320C17, such as digital volume control, noise filtering, etc. A masked ROM version of the TMS320C17 can provide a cost-effective solution.

Figure 6–21. Answering Machine



6.7 TMS320C14 System Applications

The TMS320C14 has been designed for a wide range of applications in digital signal processing and digital control. A large number of on-chip peripherals have been integrated into the TMS320C14, giving it direct-connect capability with various external devices. This can reduce and even eliminate interfacing hardware.

The following buses, ports, and control signals provide system interfacing to the TMS320C14.

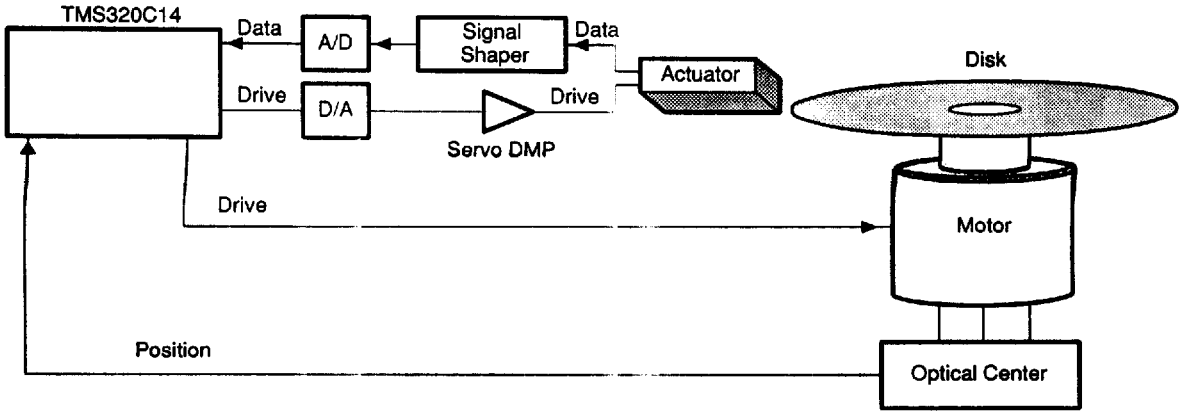
- 12-bit address bus (A11– A0)
- 16-bit data bus (D15–D0)
- 3-bit port address bus (PA2–PA0)
- Enable signals; read enable ($\overline{REN}$) and write enable ($\overline{WE}$)
- Memory control and nonmaskable interrupt signals ($\overline{NMI}$/MC/$\overline{MP}$) signals.
- Interrupt ($\overline{INT}$)
- 16 bit I/O pins (IOP15–IOP0)
- Serial port (RXD, TXD, TCLK1, TCLK2, CMP4/CAP2/FSR, CMP5/CAP3/FSX)
- Timers (WDT, TCLK/CLKR, TCLK2/CLKX)
- Event manager (CMP0–CMP3, CAP0–CAP1, CMP4/CAP2/FSR, CMP5/CAP3/FSX)
- Reset ($\overline{RS}$)

The fast CPU and associated circuitry of the TMS320C14 make it effective for computation-intensive applications requiring processing of signals in control systems, speech systems, telecommunications, FFTs, and filtering systems. The TMS320C14 surpasses previous DSPs with its large amount of peripheral circuitry. These integrated modules increase cost effectiveness by reducing (and possibly eliminating) the amount of glue circuitry normally required with other DSPs.

6.7.1 Disk Drive Control

The implementation of the TMS320C14 for disk-drive control is shown in Figure 6–22. In this example, both the read/write actuator and the spindle motor are controlled by the DSP.

Figure 6–22. Disk Drive Control



Using sophisticated algorithms, the TMS320C14 can perform the following functions:

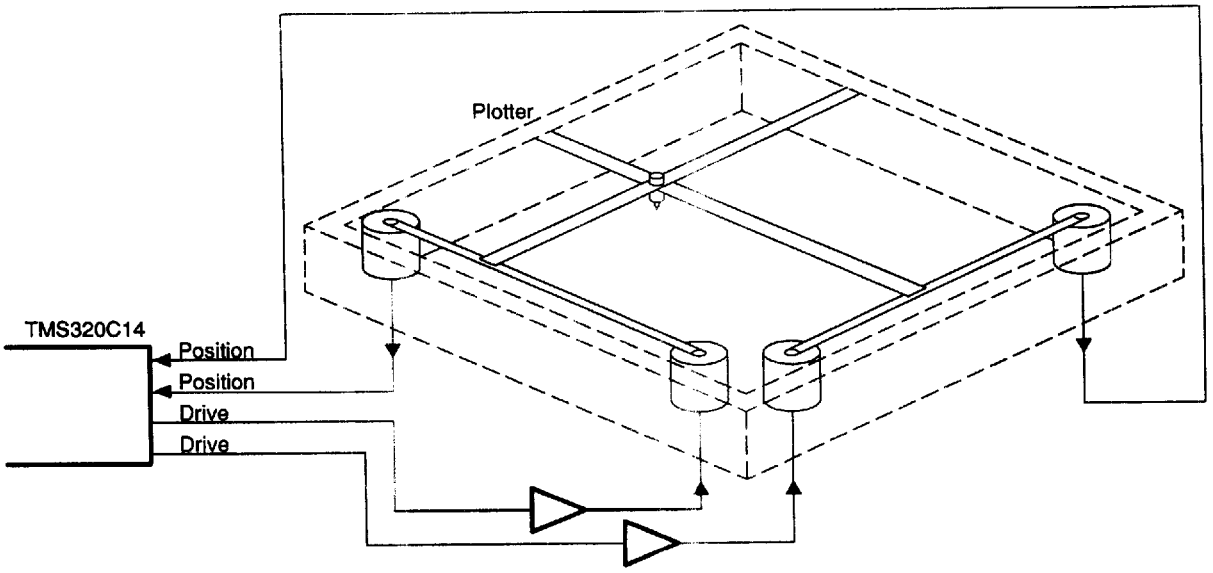
- Precisely control the read/write actuator for fast access time.
- Cancel mechanical resonance.
- Estimate and control actuator speed.

Position information is read from the disk surface and converted into digital form by an A/D converter. A D/A converter provides the position control signal for the actuator. The PWM outputs control the speed of the spindle motor directly, while the capture inputs accept the optical encoder signals indicating velocity

6.7.2 Plotter Control

Figure 6–23 shows the TMS320C14 used as a pen position controller for a plotter. The capture inputs receive position signals from the optical encoders of the the position rotors, and the PWM outputs provide the controlling signals to the X-Y drive motors.

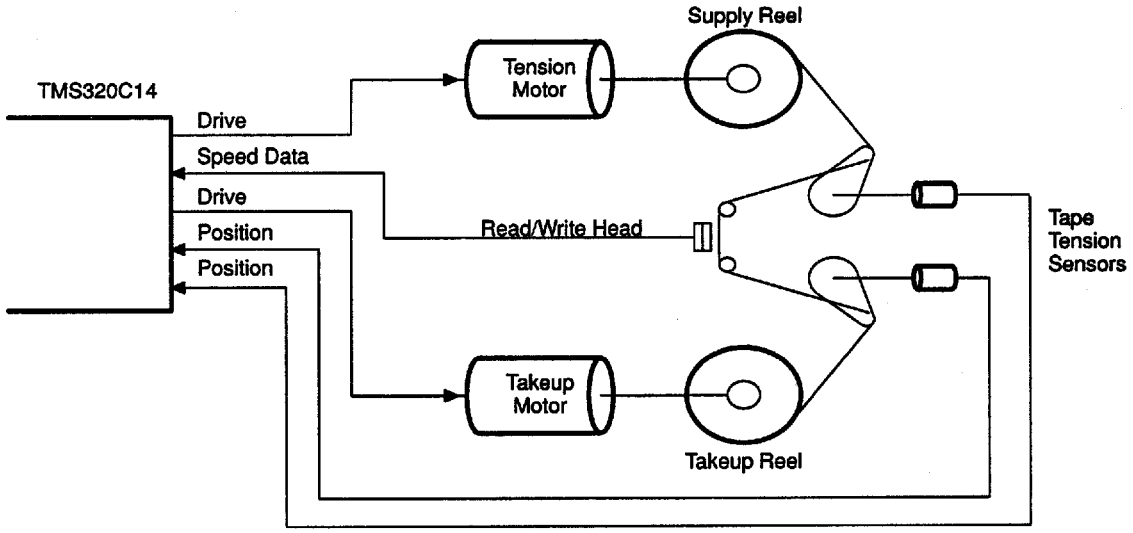
Figure 6–23. Plotter Control



6.7.3 Tape Drive Control

Figure 6–24 shows the TMS320C14 providing servo control of a magnetic tape drive mechanism. The PWM outputs generate the signals for controlling the tension and takeup motors, while the optical encoder sensors provide input to the capture inputs. The DSP is responsible for keeping the speed of the tape constant, as well as maintaining proper tape tension. Tape speed may also be detected by reading embedded information from the tape. Tape tension is detected by the position of lever arms attached to servo motors.

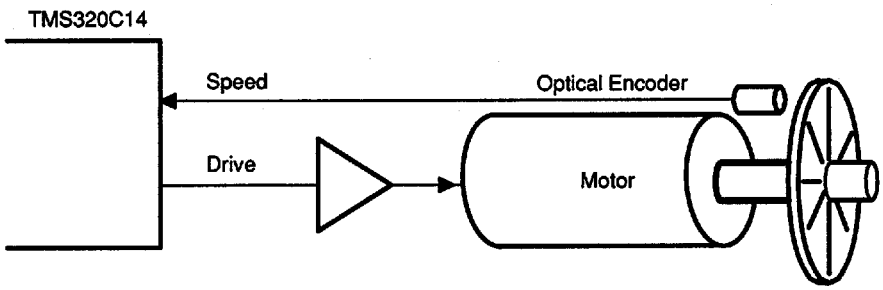
Figure 6–24. Tape Drive Control



6.7.4 AC Motor Control

The TMS320C14 provides cost-effective servo control of either AC induction and synchronous motors or DC brushless motors. AC motors are cheaper to manufacture and easier to maintain than DC servo motors. Their control structure, however, is much more complex than DC servo motors. Vector rotation techniques are used to transform the coordinates and simplify the control structure of an AC motor similarly to a field-controlled DC motor. The TMS320C14 perform the necessary computations that permit vector control of AC motors. The PWM outputs can control the multiple phases. Speed information can be obtained from optical encoders. Figure 6–25 shows an implementation of control of an AC motor.

Figure 6–25. AC Motor Control



Appendix A

TMS320C1x Digital Signal Processors

This appendix contains electrical specifications,timing, and mechanical data for all TMS320C1x devices.

JANUARY 1987 — REVISED JULY 1991

- Performance Up to 8.77 MIPs
- All TMS320C1x Devices are Object Code Compatible
- 144/256-Word On-Chip Data RAM
- 1.5K/4K/8K-Word On-Chip Program ROM
- 4K-Word On-Chip Program EPROM (TMS320E14/P14/E15/P15/E17/P17)
- One-Time Programmable (OTP) Versions Available (TMS320P14/P15/P17)
- EPROM Code Protection for Copyright Security
- 4K / 64K-Word Total External Memory at Full Speed
- 32-Bit ALU/Accumulator
- 16 × 16-Bit Multiplier With a 32-Bit Product
- 0 to 16-Bit Barrel Shifter
- Eight Input/Output Channels
- Dual-Channel Serial Port
- Simple Memory and I/O Interface
- 5-V and 3.3-V Versions Available (TMS320LC15/LC17)
- Commercial and Military Versions Available
- Operating Free-Air Temperature . . . 0°C to 70°C
- Packaging: DIP, PLCC, Quad Flatpack, and CER-QUAD
- CMOS Technology:

Device	Cycle Time
— TMS320C10	200-ns
— TMS320C10-14	280-ns
— TMS320C10-25	160-ns
— TMS320C14	160-ns
— TMS320E14	160-ns
— TMS320P14	160-ns
— TMS320C15	200-ns
— TMS320C15-25	160-ns
— TMS320E15	200-ns
— TMS320E15-25	160-ns
— TMS320LC15	250-ns
— TMS320P15	200-ns
— TMS320C16	114-ns
— TMS320C17	200-ns
— TMS320E17	200-ns
— TMS320LC17	278-ns
— TMS320P17	200-ns

Introduction

The TMS32010 digital signal processor (DSP), introduced in 1983, was the first DSP in the TMS320 family. From it has evolved this TMS320C1x generation of 16-bit DSPs. All 'C1x DSPs are object code compatible with the TMS32010 DSP. The C1x DSPs combine the flexibility of a high-speed controller with the numerical capability of an array processor thereby offering an inexpensive alternative to multichip bit-slice processors. The highly paralleled architecture and efficient instruction set provide speed and flexibility to produce a CMOS microprocessor generation capable of executing up to 8.77 MIPS (million instructions per second) ('C16). These 'C1x devices utilize a modified Harvard architecture to optimize speed and flexibility, implementing functions in hardware that other processors implement through microcode or software.

The 'C1x generation's powerful instruction set, inherent flexibility, high-speed number-handling capabilities, reduced power consumption, and innovative architecture have made these cost-effective DSPs the ideal solution for many telecommunications, computer, commercial, industrial, and military applications.

This data sheet provides detailed design documentation for the 'C1x DSPs. It facilitates the selection of devices best suited for various user applications by providing specifications and special features for each 'C1x DSP.

This data sheet is arranged as follows: introduction, quick reference table of device parameters and packages, summary overview of each device, architecture overview, and the 'C1x device instruction set summary. These are followed by data sheets for each 'C1x device providing available package styles, terminal function tables, block diagrams and electrical and timing parameters. An index is provided to facilitate data sheet usage.

Table 1 provides an overview of 'C1x processors with comparisons of memory, I/O, cycle timing, military support, and package types. For specific availability, contact the nearest TI Field Sales Office.

Table 1. TMS320C1x Device Overview

DEVICE	MEMORY				I/O		CYCLE	PACKAGE (1)		
	RAM	ROM	EPROM	PROG.	SERIAL	PARALLEL	(ns)	DIP	PLCC	CER-QUAD
TMS320C10 (2)	144	1.5K	—	4K	—	8 × 16	200	40	44	—
TMS320C10-14	144	1.5K	—	4K	—	8 × 16	280	40	44	—
TMS320C10-25	144	1.5K	—	4K	—	8 × 16	160	40	44	—
TMS320C14 (3)	256	4K	—	4K	1	7 × 16 (4)	160	—	68	—
TMS320E14 (3)	256	—	4K	4K	1	7 × 16 (4)	160	—	—	68 CER
TMS320P14†	256	—	4K	4K	1	7 × 16 (4)	160	—	68	—
TMS320C15 (3)	256	4K	—	4K	—	8 × 16	200	40	44	—
TMS320C15-25	256	4K	—	4K	—	8 × 16	160	40	44	—
TMS320E15 (3)	256	—	4K	4K	—	8 × 16	200	40	—	44 CER
TMS320E15-25	256	—	4K	4K	—	8 × 16	160	40	—	44 CER
TMS320LC15	256	4K	—	4K	—	8 × 16	250	40	44	—
TMS320P15†	256	—	4K	4K	—	8 × 16	200	40	44	—
TMS320C16	256	8K	—	64K	—	8 × 16	114	—	—	64 QFP
TMS320C17	256	4K	—	—	2	6 × 16 (5)	200	40	44	—
TMS320E17 (5)	256	—	4K	—	2	6 × 16 (5)	200	40	—	44 CER
TMS320LC17 (5)	256	4K	—	—	2	6 × 16 (5)	278	40	44	—
TMS320P17 (5)†	256	—	4K	—	2	6 × 16 (5)	200	40	44	

† One-time programmable (OTP) device is in a windowless plastic package and cannot be erased.

NOTES: 1. DIP = dual in-line package. PLCC = plastic-leaded chip carrier. CER = ceramic-leaded chip carrier. QFP = plastic quad flat pack.
2. Military version available.
3. Military versions planned; contact nearest TI Field Sales Office for availability.
4. On-chip 16-bit I/O, four capture inputs, and six compare outputs are available.
5. On-chip 16-bit coprocessor interface is optional by pin selection.

description

TMS320C10

The 'C10 provides the core CPU used in all other 'C1x devices. Its microprocessor operates at 5 MIPS. It provides a parallel I/O of 8 × 16 bits. Three versions with cycle times of 160, 200, and 280 ns are available as illustrated in Table 1. The 'C10 versions are offered in plastic 40-pin DIP or a 44-lead PLCC packages.

TMS320C14/E14/P14

The 'C14/E14/P14 devices. using the 'C10 core CPU, offer expanded on-chip RAM, and ROM or EPROM ('E14/P14), 16 pins of bit selectable parallel I/O, an I/O mapped asynchronous serial port, four 16-bit timers, and external/internal interrupts. The 'C14 devices can provide for microcomputer/microprocessor operating modes. Three versions with cycle times of 160-ns are available as illustrated in Table 1. These devices are offered in 68-pin plastic PLCC or ceramic CER-QUAD packages.

TMS320C15/E15/P15

The 'C15/E15/P15 devices are a version of the 'C10, offering expanded on-chip RAM, and ROM or EPROM ('E15/P15). The 'P15 is a one-time programmable (OTP), windowless EPROM version. These devices can operate in the microcomputer or microprocessor modes. Five versions are available with cycle times of 160 to 200 ns (see Table 1). These devices are offered in 40-pin DIP, 44-pin PLCC, or 44-pin ceramic packages.

TMS320LC15

The 'LC15 is a low-power version of the 'C15, utilizing a V_{DD} of only 3.3-V. This feature results in a 2.3: 1 power requirement reduction over the typical 5-V 'C1x device. It operates at a cycle time of 250 ns. The device is offered in 40-pin DIP or 44-lead PLCC packages.

TMS320C16

The 'C16 offers on-chip RAM of 256-words, an expanded program memory of 64K-words, and a fast instruction cycle time of 114 ns (8.77 MIPS). It is offered in a 64-pin quad flat-pack package.

TMS320C17/E17/P17

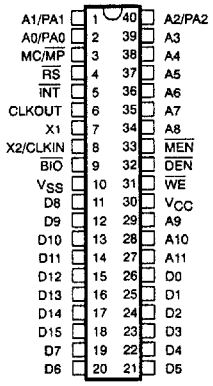
The 'C17/E17/P17 versions consist of five major functional units: the 'C15 microcomputer, a system control register, a full-duplex dual channel serial port, μ-law/A-law companding hardware, and a coprocessor port. The dual-channel serial port is capable of full-duplex serial communication and offers direct interface to two combo-codecs. The hardware companding logic can operate in either μ-law or A-law format with either sign-magnitude or twos complement numbers in either serial or parallel modes. The coprocessor port allows the 'C17/E17/P17 to act as a slave microcomputer or as a master to a peripheral microcomputer.

The 'P17 utilizes a one-time programmable (OTP) windowless EPROM version of the 'E17.

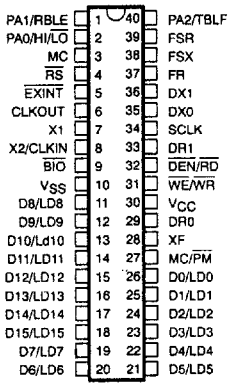
TMS320LC17

The 'LC17 is a low-power version of the 'C17, utilizing a V_{DD} of only 3.3-V. This feature results in a 2.3: 1 power requirement reduction over the typical 5-V 'C1x device. It operates at a cycle time of 278 ns.

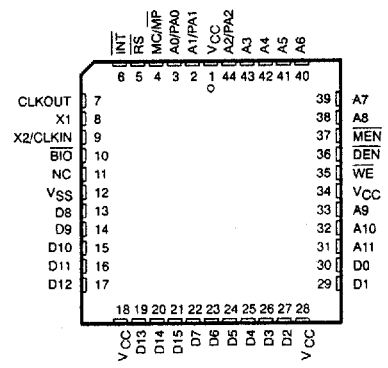
TMS320C10/C15/LC15/P15
N/JD Packages
(Top View)



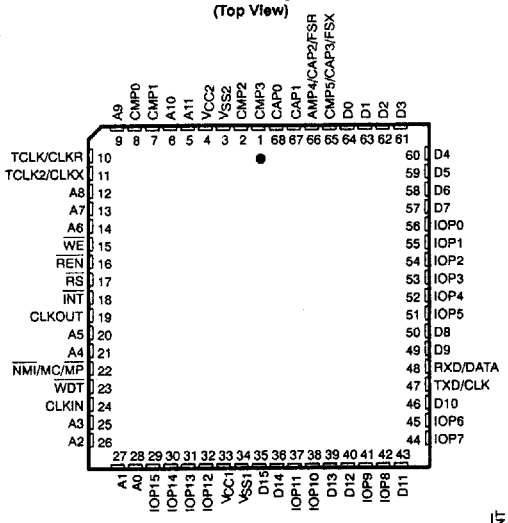
TMS320C17/E17/LC17/P17
N/JD Packages
(Top View)



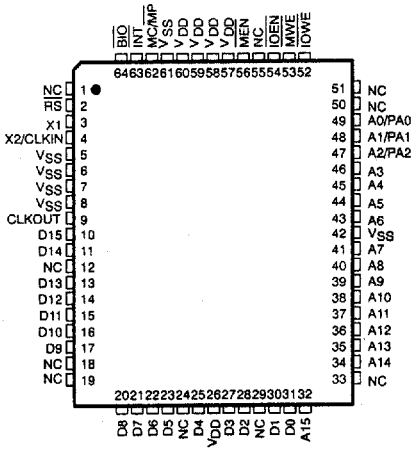
TMS320C10/C15/E15/LC15/P15
FN/FZ Packages
(Top View)



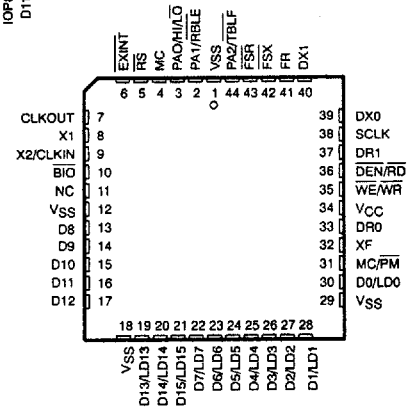
TMS320C14/E14/P14
FN/FZ Packages
(Top View)



TMS320C16
PG Package
(Top View)



TMS320C17/E17
FN/FZ Packages
(Top View)



architecture

The 'C1x DSPs use a modified Harvard architecture for speed and flexibility. In a strict Harvard architecture, program and data memory lie in two separate spaces, permitting a full overlap of instruction fetch and one-cycle execution. The 'C1x DSPs modification allows transfers between program and data spaces, thereby increasing the flexibility of the device. This modification permits coefficients stored in program memory to be read into the RAM, eliminating the need for a separate coefficient ROM.

32-bit accumulator

All 'C1x devices contain a 32-bit ALU and accumulator for support of double-precision, twos-complement arithmetic. The ALU is a general-purpose arithmetic unit that operates on 16-bit words taken from the data RAM or derived from immediate instructions. In addition to the usual arithmetic instructions, the ALU can perform Boolean operations, providing the bit manipulation ability required of a high-speed controller. The accumulator stores the output from the ALU and is often an input to the ALU. It operates with a 32-bit word length. The accumulator is divided into a high-order word (bits 31 through 16) and a low-order word (bits 15 through 0). Instructions are provided for storing the high- and low-order accumulator words in memory.

shifters

Two shifters are available for manipulating data. The ALU barrel shifter performs a left-shift of 0 to 16 places on data memory words loaded into the ALU. This shifter extends the high-order bit of the data word and zero-fills the low-order bits for twos-complement arithmetic. The accumulator parallel shifter performs a left-shift of 0, 1 or 4 places on the entire accumulator and places the resulting high-order accumulator bits into data RAM. Both shifters are useful for scaling and bit extraction.

16 × 16-bit parallel multiplier

The multiplier performs a 16 × 16-bit twos-complement multiplication with a 32-bit result in a single instruction cycle. The multiplier consists of three units: the T Register, P Register, and a multiplier array. The 16-bit T Register stores the multiplicand, and the P Register stores the 32-bit product. Multiplier values either come from the data memory or are derived immediately from the MPYK (multiply immediate) instruction word. The fast on-chip multiplier allows the device to perform fundamental operations such as convolution, correlation, and filtering.

data and program memory

Since the 'C1x devices use a Harvard type architecture, data and program memory reside in two separate spaces. These DSP devices have 144-or 256-words of on-chip data RAM and 1.5K- to 8K-words of on-chip program ROM. On-chip program EPROM of 4K-words is provided in the 'E14/E15/E17 devices. An on-chip one-time programmable 4K-word EPROM is provided in the 'P14/P15/P17 devices. The EPROM cell utilizes standard PROM programmers and is programmed identically to a 64K CMOS EPROM (TMS27C64). (Reference Table 1.)

program memory expansion

All 'C1x devices except the 'C17/E17/LC17/P17 devices are capable of executing from off-chip external memory at full speed for those applications requiring external program memory space. This allows for external RAM-based systems to provide multiple functionality. The 'C17/E17/LC17/P17 devices provide no external memory expansion. (Reference Table 1.)

microcomputer/microprocessor operating modes

All devices except the 'x17 offer two modes of operation defined by the state of the MC/$\overline{MP}$ pin: the microcomputer mode (MC/$\overline{MP}$ = 1) or the microprocessor mode (MC/$\overline{MP}$ = 0). In the microcomputer mode, on-chip ROM is mapped into the program memory space. In the microprocessor mode, all words of progam memory are external.

interrupts and subroutines

All devices except the 'C16 contain a four-level stack for saving the contents of the program counter during interrupts and subroutine calls. Because of the larger 64K program space, the 'C16's hardware stack has been increased to eight levels. Instructions are available for saving the device's complete context. PUSH and POP instructions permit a level of nesting restricted only by the amount of available RAM. The interrupts used in these devices are maskable.

input/output

The 16-bit parallel data bus can be utilized to perform I/O functions in two cycles. The I/O ports are addressed by the three LSBs on the address lines. In addition, a polling input for bit test and jump operations ($\overline{BIO}$) and an interrupt pin ($\overline{INT}$) have been incorporated for multitasking. The bit selectable I/O of the 'C14 is suitable for microcontroller applications.

serial port (TMS320C17/E17)

Two of the I/O ports on the 'C17/E17 are dedicated to the serial port and companding hardware. I/O port 0 is dedicated to control register 0, which controls the serial port, interrupts, and companding hardware. I/O port 1 accesses control register 1, as well as both serial port channels, and companding hardware. The six remaining I/O ports are available for external parallel interfaces.

serial port (TMS320C14/E14)

The 'C14/E14 devices include one I/O-mapped serial port that operates asynchronously. I/O-mapped control registers are used to configure port parameters such as inter-processor communication protocols and baud rate.

companding hardware (TMS320C17/E17)

On-chip hardware enables the 'C17/E17 to compand (COMpress/exPAND) data in either μ-law or A-law format. The companding logic operation is configured via the system control register. Data may be companded in either serial mode for operation on serial port data (converting between linear and logarithmic PCM) or a parallel mode for computation inside the device. The 'C17/E17 allows the hardware companding logic to operate with either sign-magnitude or twos-complement numbers.

coprocessor port (TMS320C17/E17)

The coprocessor port on the 'C17/E17 provides a direct connection to most microcomputers and microprocessors. The port is accessed through I/O port 5 using IN and OUT instructions. The coprocessor interface allows the device to act as a peripheral (slave) microcomputer to a microprocessor, or as a master to a peripheral microcomputer. In the microcomputer mode, the 16 data lines are used for the 6 parallel 16-bit I/O ports. In the coprocessor mode, the 16-bit parallel port is reconfigured to operate as a 16-bit latched bus interface. For peripheral transfer, an 8-bit or 16-bit length of the coprocessor port can be selected.

Instruction set

A comprehensive instruction set supports both numeric-intensive operations, such as signal processing, and general-purpose operations, such as high-speed control. All of the 'C1x devices are object-code compatible and use the same 60 instructions. The instruction set consists primarily of single-cycle single-word instructions, permitting execution rates of more than six million instructions per second. Only infrequently used branch and I/O instructions are multicycle. Instructions that shift data as part of an arithmetic operation execute in a single cycle and are useful for scaling data in parallel with other operations.

NOTE

The $\overline{BIO}$ pin on other 'C1x devices is not available for use in the 'C14/E14/P14. An attempt to execute the BIOZ (Branch on $\overline{BIO}$ low) instruction will result in a two cycle NOP action.

Three main addressing modes are available with the instruction set: direct, indirect, and immediate addressing.

direct addressing

In direct addressing, seven bits of the instruction word concatenated with the 1-bit data page pointer form the data memory address. This implements a paging scheme in which the first page contains 128 words, and the second page contains up to 128 words.

indirect addressing

Indirect addressing forms the data memory address from the least-significant eight bits of one of the two auxiliary registers, AR0-AR1. The Auxiliary Register Pointer (ARP) selects the current auxiliary register. The auxiliary registers can be automatically incremented or decremented and the ARP changed in parallel with the execution of any indirect instruction to permit single-cycle manipulation of data tables. Indirect addressing can be used with all instructions requiring data operands, except for the immediate operand instructions.

immediate addressing

Immediate instructions derive data from part of the instruction word rather than from the data RAM. Some useful immediate instructions are multiply immediate (MPYK), load accumulator immediate (LACK), and load auxiliary register immediate (LARK).

instruction set summary

Table 2 lists the symbols and abbreviations used in Table 3, the instruction set summary. Table 3 contains a short description and the opcode for each 'C1x instruction. The summary is arranged according to function and alphabetized within each functional group.

Table 2. Instruction Symbols

SYMBOL	MEANING
ACC	Accumulator
D	Data memory address field
M	Addressing mode bit
K	Immediate operand field
PA	3-bit port address field
R	1-bit operand field specifying auxiliary register
S	4-bit left-shift code
X	3-bit accumulator left-shift field

Table 3. TMS320C1x Instruction Set Summary

ACCUMULATOR INSTRUCTIONS																			
MNEMONIC	DESCRIPTION	NO. CYCLES	NO. WORDS	OPCODE INSTRUCTION REGISTER															
				15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ABS	Absolute value of accumulator	1	1	0	1	1	1	1	1	1	1	1	0	0	0	1	0	0	0
ADD	Add to accumulator with shift	1	1	0	0	0	0	◄		S	►	M	◄		D			►	
ADDH	Add to high-order accumulator bits	1	1	0	1	1	0	0	0	0	0	M	◄		D			►	
ADDS	Add to accumulator with no sign extension	1	1	0	1	1	0	0	0	0	1	M	◄		D			►	
AND	AND with accumulator	1	1	0	1	1	1	1	0	0	1	M	◄		D			►	
LAC	Load accumulator with shift	1	1	0	0	1	0	◄		S	►	M	◄		D			►	
LACK	Load accumulator immediate	1	1	0	1	1	1	1	1	1	0	◄			K			►	
OR	OR with accumulator	1	1	0	1	1	1	1	0	1	0	M	◄		D			►	
SACH	Store high-order accumulator bits with shift	1	1	0	1	0	1	1	◄	X	►	M	◄		D			►	
SACL	Store low-order accumulator bits	1	1	0	1	0	1	0	0	0	0	M	◄		D			►	
SUB	Subtract from accumulator with shift	1	1	0	0	0	1	◄		S	►	M	◄		D			►	
SUBC	Conditional subtract (for divide)	1	1	0	1	1	0	0	1	0	0	M	◄		D			►	
SUBH	Subtract from high-order accumulator bits	1	1	0	1	1	0	0	0	1	0	M	◄		D			►	
SUBS	Subtract from accumulator with no sign extension	1	1	0	1	1	0	0	0	1	1	M	◄		D			►	
XOR	Exclusive OR with accumulator	1	1	0	1	1	1	1	0	0	0	M	◄		D			►	
ZAC	Zero accumulator	1	1	0	1	1	1	1	1	1	1	1	0	0	0	1	0	0	1
ZALH	Zero accumulator and load high-order bits	1	1	0	1	1	0	0	1	0	1	M	◄		D			►	
ZALS	Zero accumulator and load low-order bits with no sign extension	1	1	0	1	1	0	0	1	1	0	M	◄		D			►	

AUXILIARY REGISTER AND DATA PAGE POINTER INSTRUCTIONS																			
MNEMONIC	DESCRIPTION	NO. CYCLES	NO. WORDS	OPCODE INSTRUCTION REGISTER															
				15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
LAR	Load auxiliary register	1	1	0	0	1	1	1	0	0	R	M	◄		D			►	
LARK	Load auxiliary register immediate	1	1	0	1	1	1	0	0	0	R	◄			K			►	
LARP	Load auxiliary register pointer immediate	1	1	0	1	1	0	1	0	0	0	1	0	0	0	0	0	0	K
LDP	Load data memory page pointer	1	1	0	1	1	0	1	1	1	1	M	◄		D			►	
LDPK	Load data memory page pointer immediate	1	1	0	1	1	0	1	1	1	0	0	0	0	0	0	0	0	K
MAR	Modify auxiliary register and pointer	1	1	0	1	1	0	1	0	0	0	M	◄		D			►	
SAR	Store auxiliary register	1	1	0	0	1	1	0	0	0	R	M	◄		D			►	

Table 3. TMS320C1x Instruction Set Summary (continued)

BRANCH INSTRUCTIONS																	
MNEMONIC	DESCRIPTION	NO. CYCLES	NO. WORDS	OPCODE INSTRUCTION REGISTER													
				15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
B	Branch unconditionally	2	2	1	1	1	1	1	0	0	1	0	0	0	0	0	0	0	0
				0	0	0	0	← BRANCH ADDRESS →											
BANZ	Branch on auxiliary register not zero	2	2	1	1	1	1	0	1	0	0	0	0	0	0	0	0	0	0
				0	0	0	0	← BRANCH ADDRESS →											
BGEZ	Branch if accumulator ≥ 0	2	2	1	1	1	1	1	1	0	1	0	0	0	0	0	0	0	0
				0	0	0	0	← BRANCH ADDRESS →											
BGZ	Branch if accumulator > 0	2	2	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0
				0	0	0	0	← BRANCH ADDRESS →											
BIOZ	Branch on $\overline{BIO}$ = 0 †	2	2	1	1	1	1	0	1	1	0	0	0	0	0	0	0	0	0
				0	0	0	0	← BRANCH ADDRESS →											
BLEZ	Branch if accumulator ≤ 0	2	2	1	1	1	1	1	0	1	1	0	0	0	0	0	0	0	0
				0	0	0	0	← BRANCH ADDRESS →											
BLZ	Branch if accumulator < 0	2	2	1	1	1	1	1	0	1	0	0	0	0	0	0	0	0	0
				0	0	0	0	← BRANCH ADDRESS →											
BNZ	Branch if accumulator ≠ 0	2	2	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0
				0	0	0	0	← BRANCH ADDRESS →											
BV	Branch on overflow	2	2	1	1	1	1	0	1	0	1	0	0	0	0	0	0	0	0
				0	0	0	0	← BRANCH ADDRESS →											
BZ	Branch if accumulator = 0	2	2	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0
				0	0	0	0	← BRANCH ADDRESS →											
CALA	Call subroutine from accumulator	2	1	0	1	1	1	1	1	1	1	1	0	0	0	1	1	0	0
CALL	Call subroutine immediately	2	2	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0
				0	0	0	0	← BRANCH ADDRESS →											
RET	Return from subroutine or interrupt routine	2	1	0	1	1	1	1	1	1	1	1	0	0	0	1	1	0	1

T REGISTER, P REGISTER, AND MULTIPLY INSTRUCTIONS																	
MNEMONIC	DESCRIPTION	NO. CYCLES	NO. WORDS	OPCODE INSTRUCTION REGISTER													
				15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
APAC	Add P register to accumulator	1	1	0	1	1	1	1	1	1	1	1	0	0	0	1	1	1	1
LT	Load T Register	1	1	0	1	1	0	1	0	1	0	M	← D →						
LTA	LTA combines LT and APAC into one instruction	1	1	0	1	1	0	1	1	0	0	M	← D →						
LTD	LTD combines LT, APAC, and DMOV into one instruction	1	1	0	1	1	0	1	0	1	1	M	← D →						
MPY	Multiply with T register, store product in P register	1	1	0	1	1	0	1	1	0	1	M	← D →						
MPYK	Multiply T register with immediate operand; store product in P register	1	1	1	0	0	← K →												
PAC	Load accumulator from P register	1	1	0	1	1	1	1	1	1	1	1	0	0	0	1	1	1	0
SPAC	Subtract P register from accumulator	1	1	0	1	1	1	1	1	1	1	1	0	0	1	0	0	0	0

† This instruction is a NOP on the '320C14/E14/P14.

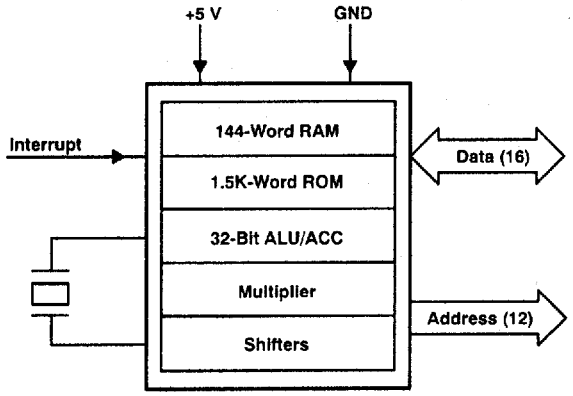
Table 3. TMS320C1x Instruction Set Summary (concluded)

CONTROL INSTRUCTIONS																	
MNEMONIC	DESCRIPTION	NO. CYCLES	NO. WORDS	OPCODE INSTRUCTION REGISTER													
				15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DINT	Disable interrupt	1	1	0	1	1	1	1	1	1	1	1	0	0	0	0	0	0	1
EINT	Enable interrupt	1	1	0	1	1	1	1	1	1	1	1	0	0	0	0	0	1	0
LST	Load status register	1	1	0	1	1	1	1	0	1	1	M	◄				D		►
NOP	No operation	1	1	0	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0
POP	POP stack to accumulator	2	1	0	1	1	1	1	1	1	1	1	0	0	1	1	1	0	1
PUSH	PUSH stack from accumulator	2	1	0	1	1	1	1	1	1	1	1	0	0	1	1	1	0	0
ROVM	Reset overflow mode	1	1	0	1	1	1	1	1	1	1	1	0	0	0	1	0	1	0
SOVM	Set overflow mode	1	1	0	1	1	1	1	1	1	1	1	0	0	0	1	0	1	1
SST	Store status register	1	1	0	1	1	1	1	1	0	0	M	◄				D		►

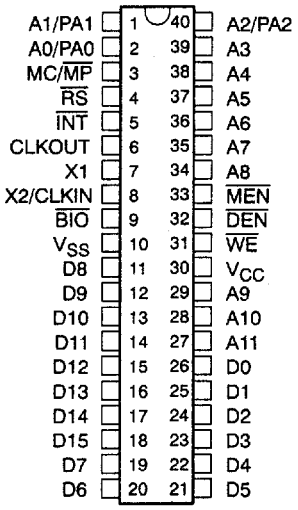
I/O AND DATA MEMORY OPERATIONS																	
MNEMONIC	DESCRIPTION	NO. CYCLES	NO. WORDS	OPCODE INSTRUCTION REGISTER													
				15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DMOV	Copy contents of data memory location into next higher location	1	1	0	1	1	0	1	0	0	1	M	◄				D		►
IN	Input data from port	2	1	0	1	0	0	0	◄	PA	►	M	◄				D		►
OUT	Output data to port	2	1	0	1	0	0	1	◄	PA	►	M	◄				D		►
TBLR	Table read from program memory to data RAM	3	1	0	1	1	0	0	1	1	1	M	◄				D		►
TBLW	Table write from data RAM to program memory	3	1	0	1	1	1	1	1	0	1	M	◄				D		►

Key Features: TMS320C10

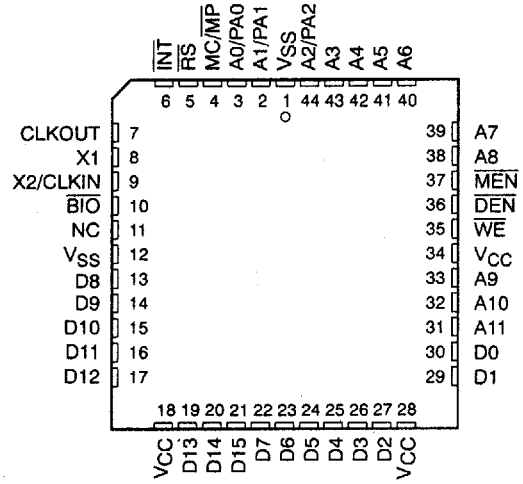
- Instruction Cycle Timing
  - 160-ns (TMS320C10-25)
  - 200-ns (TMS32010)
  - 280-ns (TMS320C10-14)
- 144 Words of On-Chip Data RAM
- 1.5K Words On-Chip Program ROM
- External Memory Expansion up to 4K Words at Full Speed
- 16 × 16-Bit Multiplier With 32-Bit Product
- 0 to 16-Bit Barrel Shifter
- On-Chip Clock Oscillator
- Device Packaging:
  - 40-Pin DIP
  - 44-Lead PLCC
- Single 5-V Supply
- Operating Free-Air Temperature Range . . . 0°C to 70°C



TMS320C10
N/JD Package
(Top View)



TMS320C10
FN/FZ Package
(Top View)

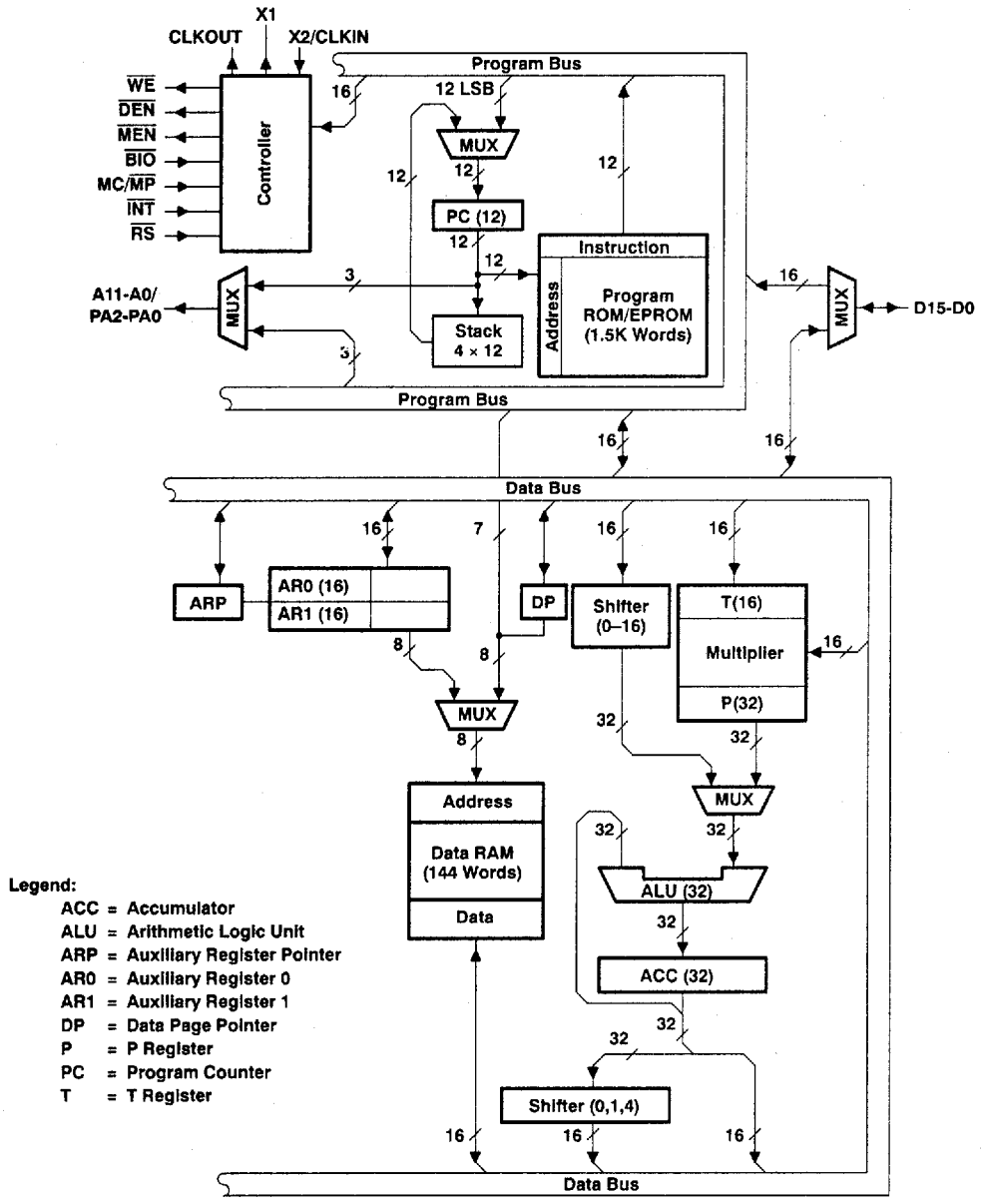


TERMINAL FUNCTIONS

NAME	I/O†	DEFINITION
A11-A0/PA2-PA0	O	External address bus. I/O port address multiplexed over PA2-PA0.
$\overline{BIO}$	I	External polling input
CLKOUT	O	System clock output, 1/4 crystal/CLKIN frequency
D15-D0	I/O	16-bit parallel data bus
$\overline{DEN}$	O	Data enable for device input data on D15-D0
$\overline{INT}$	I	External interrupt input
MC/$\overline{MP}$	I	Memory mode select pin. High selects microcomputer mode. Low selects microprocessor mode.
$\overline{MEN}$	O	Memory enable indicates that D15-D0 will accept external memory instruction.
NC	O	No connection
$\overline{RS}$	I	Reset for initializing the device
V_{CC}	I	+ 5 V supply
V_{SS}	I	Ground
$\overline{WE}$	O	Write enable for device output data on D15-D0
X1	O	Crystal output for internal oscillator
X2/CLKIN	I	Crystal input internal oscillator or external system clock input

† Input/Output/High-impedance state.

functional block diagram



electrical specifications

This section contains the electrical specifications for all speed versions of the 'C10 Digital Signal Processors, including test parameter measurement information.

absolute maximum ratings over operating free-air temperature range (unless otherwise noted)†

Supply voltage range V_{CC} (see Note 6) ..... −0.3 V to 7 V
Input voltage range ..... −0.3 V to 7 V
Output voltage range ..... −0.3 V to 7 V
Continuous power dissipation ..... 0.5 mW
Operating free-air temperature: L suffix ..... 0°C to 70°C
A suffix ..... −40°C to 85°C
Storage temperature ..... −55 °C to 150 °C

† Stresses beyond those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. This is a stress rating only, and functional operation of the device at these or any other conditions beyond those indicated in the "Recommended Operating Conditions" section of this specification is not implied. Exposure to absolute-maximum-rated conditions for extended periods may affect device reliability.

NOTE 6: All voltage values are with respect to V_{SS}.

recommended operating conditions

			MIN	NOM	MAX	UNIT
V_{CC}	Supply voltage		4.5	5	5.25	V
V_{SS}	Supply voltage			0		V
V_{IH}	High-level input voltage	CLKIN	3			V
		All remaining inputs	2			V
V_{IL}	Low-level input voltage	MC/$\overline{MP}$			0.6	V
		All remaining inputs			0.8	V
I_{OH}	High-level output current, all outputs				−300	μA
I_{OL}	Low-level output current				2	mA
T_A	Operating free-air temperature	L suffix	0		70	°C
		A suffix	−40		85	°C

electrical characteristics over specified temperature range (unless otherwise noted)

PARAMETER			TEST CONDITIONS		MIN	TYP†	MAX	UNIT
V_{OH}	High-level output voltage		I_{OH} = MAX		2.4	3		V
			I_{OH} = 20 μA (see Note 7)		V_{CC} – 0.4‡			
V_{OL}	Low-level output voltage		I_{OL} = MAX			0.3	0.5	V
I_{OZ}	Off-state output current		V_{CC} = MAX	V_O = 2.4 V			20	μA
				V_O = 0.4 V			– 20	
I_I	Input current		V_{CC} = V_{SS} to V_{CC}	All inputs except CLKIN			±20	μA
				CLKIN			±50	
C_i	Input capacitance	Data bus	f = 1 MHz, all other pins 0 V			25‡		pF
		All others				15‡		
C_o	Output capacitance	Data bus				25‡		pF
		All others				10‡		

† All typical values are at V_{CC} = 5 V, T_A = 25°C.
‡ Values derived from characterization data and not tested.
NOTE 7: This voltage specification is included for interface to HC logic. However, note that all of the other timing parameters defined in this data sheet are specified for TTL logic levels and will differ for HC logic levels.

INTERNAL CLOCK OPTION

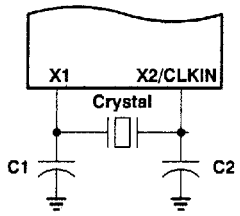


Figure 1. Internal Clock Option

PARAMETER MEASUREMENT INFORMATION

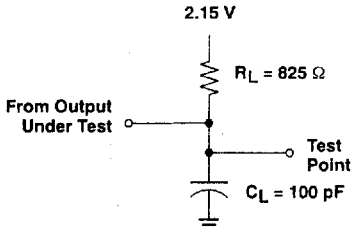


Figure 2. Test Load Circuit

electrical characteristics over specified temperature range (unless otherwise noted)

PARAMETER		TEST CONDITIONS (SEE FIGURE 2)	MIN	TYP†	MAX	UNIT
I_{CC}‡ Supply current	TMS320C10	f = 20.5 MHz, V_{CC} = 5.5 V, T_A = – 40°C to 85°C		33	55	mA
	TMS320C10-25	f = 25.6 MHz, V_{CC} = 5.5 V T_A = – 0°C to 70°C		40	65	

† All typical values are at T_A = 70°C and are used for thermal resistance calculations.
‡ I_{CC} characteristics are inversely proportional to temperature. For I_{CC} dependence on temperature, frequency, and loading.

CLOCK CHARACTERISTICS AND TIMING

The 'C10/C10-25 can use either its internal oscillator or an external frequency source for a clock.

internal clock option

The internal oscillator is enabled by connecting a crystal across X1 and X2/CLKIN (see Figure 1). The frequency of CLKOUT is one-fourth the crystal fundamental frequency. The crystal should be fundamental mode, and parallel resonant, with an effective series resistance of 30 ohms, a power dissipation of 1 mW, and should be specified at a load capacitance of 20 pF.

PARAMETER		TEST CONDITIONS	MIN	NOM	MAX	UNIT
Crystal frequency, f_x	TMS320C10	T_A = – 40°C to 85°C	6.7		20.5	MHz
	TMS320C10-25	T_A = 0°C to 70°C	6.7		25.6	
C1, C2		T_A = – 40°C to 85°C		10		pF

external clock option

An external frequency source can be used by injecting the frequency directly into X2/CLKIN with X1 left unconnected. The external frequency injected must conform to the specifications listed in the table below.

switching characteristics over recommended operating conditions

PARAMETER		TEST CONDITIONS	TMS320C10			TMS320C10-25			UNIT
			MIN	NOM	MAX	MIN	NOM	MAX	
$t_{c(C)}$	CLKOUT cycle time§	R_L = 825 Ω, C_L = 100 pF (see Figure 2)	195.12	200		156.25	160		ns
$t_{r(C)}$	CLKOUT rise time			10¶			10¶		ns
$t_{f(C)}$	CLKOUT fall time			8¶			8¶		ns
$t_{w(CL)}$	Pulse duration, CLKOUT low			92¶			72¶		ns
$t_{w(CH)}$	Pulse duration, CLKOUT high			90¶			70¶		ns
$t_{d(MCC)}$	Delay time, CLKIN↑ to CLKOUT↓		25¶		60¶	25		50¶	ns

§ $t_{c(C)}$ is the cycle time of CLKOUT, i.e., $4t_{c(MC)}$ (4 times CLKIN cycle time if an external oscillator is used).
¶ Values derived from characterization data and not tested.

timing requirements over recommended operating conditions

		TMS320C10			TMS320C10-25			UNIT
		MIN	NOM	MAX	MIN	NOM	MAX	
$t_{c(MC)}$	Master clock cycle time	48.78	50	150	39.06	40	150¶	ns
$t_{r(MC)}$	Rise time, master clock input		5¶	10¶		5¶	10¶	ns
$t_{f(MC)}$	Fall time, master clock input		5¶	10¶		5¶	10¶	ns
$t_{w(MCP)}$	Pulse duration, master clock	$0.4t_{c(MC)}$¶		$0.6t_{c(MC)}$¶	$0.45t_{c(MC)}$¶		$0.55t_{c(MC)}$¶	ns
$t_{w(MCL)}$	Pulse duration, master clock low		20¶			15¶		ns
$t_{w(MCH)}$	Pulse duration, master clock high		20¶			15¶		ns

¶ Values derived from characterization data and not tested.

MEMORY AND PERIPHERAL INTERFACE TIMING

switching characteristics over recommended operating conditions

PARAMETER		TEST CONDITIONS	TMS320C10			TMS320C10-25			UNIT
			MIN	TYP	MAX	MIN	TYP	MAX	
t_{d1}	Delay time, CLKOUT↓ to address bus valid	R_L = 825 Ω, C_L = 100 pF, (see Figure 2)	10†		50	10†		40	ns
t_{d2}	Delay time, CLKOUT↓ to $\overline{MEN}$↓		$1/4t_{c(C)} - 5$†		$1/4t_{c(C)} + 15$	$1/4t_{c(C)} - 5$†		$1/4t_{c(C)} + 12$	ns
t_{d3}	Delay time, CLKOUT↓ to $\overline{MEN}$↑		−10†		15	−10†		12	ns
t_{d4}	Delay time, CLKOUT↓ to $\overline{DEN}$↓		$1/4t_{c(C)} - 5$†		$1/4t_{c(C)} + 15$	$1/4t_{c(C)} - 5$†		$1/4t_{c(C)} + 12$	ns
t_{d5}	Delay time, CLKOUT↓ to $\overline{DEN}$↑		−10†		15	−10†		12	ns
t_{d6}	Delay time, CLKOUT↓ to $\overline{WE}$↓		$1/2t_{c(C)} - 5$†		$1/2t_{c(C)} + 15$	$1/2t_{c(C)} - 5$†		$1/2t_{c(C)} + 12$	ns
t_{d7}	Delay time, CLKOUT↓ to $\overline{WE}$↑		−10†		15	−10†		12	ns
t_{d8}	Delay time, CLKOUT↓ to data bus OUT valid				$1/4t_{c(C)} + 65$			$1/4t_{c(C)} + 52$†	ns
t_{d9}	Time after CLKOUT↓ that data bus starts to be driven		$1/4t_{c(C)} - 5$†			$1/4t_{c(C)} - 5$†			ns
t_{d10}	Time after CLKOUT↓ that data bus stops being driven				$1/4t_{c(C)} + 40$†			$1/4t_{c(C)} + 40$†	ns
t_v	Data bus OUT valid after CLKOUT↓		$1/4t_{c(C)} - 10$			$1/4t_{c(C)} - 10$			ns
$t_{h(A\text{-}WMD)}$	Address hold time after $\overline{WE}$↑, $\overline{MEN}$↑, or $\overline{DEN}$↑ (see Note 8)		−10†			−10†			ns
$t_{su(A\text{-}MD)}$	Address bus setup time prior to $\overline{MEN}$↓ or $\overline{DEN}$↓		$1/4t_{c(C)} - 45$			$1/4t_{c(C)} - 35$			ns

† Values derived from characterization data and not tested.
NOTE 8: For interfacing I/O devices, see Figure 3.

timing requirements over recommended operating conditions

		TEST CONDITION	TMS320C10			TMS320C10-25			UNIT
			MIN	NOM	MAX	MIN	NOM	MAX	
$t_{su(D)}$	Setup time, data bus valid prior to CLKOUT↓	R_L = 825 Ω, C_L = 100 pF (see Figure 2)	50			40			ns
$t_{h(D)}$	Hold time, data bus held valid after CLKOUT↓ (see Note 9)		0			0			ns

NOTE 9: Data may be removed from the data bus upon $\overline{MEN}$↑ or $\overline{DEN}$↑ preceding CLKOUT↓.

SUGGESTED I/O DECODE CIRCUIT

The circuit shown in Figure 3 is a design example for interfacing I/O devices to the 'C10/C10-25. This circuit decodes the address for output operations using the OUT instruction. The same circuit can be used to decode input and output operations if the inverter ('ALS04) is replaced with a NAND gate and both $\overline{DEN}$ and $\overline{WE}$ are connected. Inputs and outputs can be decoded at the same port provided the output of the decoder ('AS137) is gated with the appropriate signal ($\overline{DEN}$ or $\overline{WE}$) to select read or write (using an 'ALS32). Access times can be increased when the circuit shown in Figure 3 is repeated to support IN instructions with $\overline{DEN}$ connected rather than $\overline{WE}$.

The table write (TBLW) function requires a different circuit. A detailed discussion of an example circuit for this function is described in the application report, "Interfacing External Memory to the TMS32010", published in the book, *Digital Signal Processing Applications with the TMS320 Family* (SPRA012A).

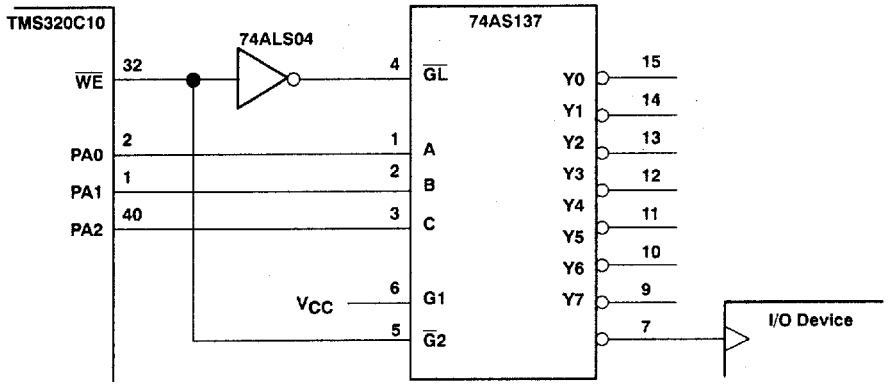


Figure 3. I/O Decode Circuit

RESET ($\overline{RS}$) TIMING

switching characteristics over recommended operating conditions

PARAMETER		TEST CONDITIONS	MIN	TYP	MAX	UNIT
t_{d11}	Delay time, $\overline{DEN}$↑, $\overline{WE}$↑, and $\overline{MEN}$↑ from $\overline{RS}$	R_L 825 Ω, C_L = 100 pF, (see Figure 2)			$1/2t_{c(C)}$ +50†	ns
$t_{dis(R)}$	Data bus disable time after $\overline{RS}$				$1/4t_{c(C)}$ +50†	ns

† Values derived from characterization data and not tested.

timing requirements over recommended operating conditions

PARAMETER		TMS320C10			TMS320C10-25			UNIT
		MIN	NOM	MAX	MIN	NOM	MAX	
$t_{su(R)}$	Reset ($\overline{RS}$) setup time prior to CLKOUT (see Note 10)	50			40			ns
$t_{w(R)}$	$\overline{RS}$ pulse duration	$5t_{c(C)}$			$5t_{c(C)}$			ns

NOTE 10: $\overline{RS}$ can occur anytime during a clock cycle. Time given is minimum to ensure synchronous operation.

INTERRUPT ($\overline{INT}$) TIMING

timing requirements over recommended operating conditions

		TMS320C10			TMS320C10-25			UNIT
		MIN	NOM	MAX	MIN	NOM	MAX	
$t_{f(INT)}$	Fall time, $\overline{INT}$			15			15	ns
$t_{w(INT)}$	Pulse duration, $\overline{INT}$	$t_{c(C)}$			$t_{c(C)}$			ns
$t_{su(INT)}$	Setup time, $\overline{INT}$↓ before CLKOUT↓	50			40			ns

IO ($\overline{BIO}$) TIMING

timing requirements over recommended operating conditions

		TMS320C10			TMS320C10-25			UNIT
		MIN	NOM	MAX	MIN	NOM	MAX	
$t_{f(IO)}$	Fall time, $\overline{BIO}$			15			15	ns
$t_{w(IO)}$	Pulse duration, $\overline{BIO}$	$t_{c(C)}$			$t_{c(C)}$			ns
$t_{su(IO)}$	Setup time, $\overline{BIO}$↓ before CLKOUT↓	50			40			ns

electrical characteristics over specified temperature range (unless otherwise noted)

PARAMETER		TEST CONDITIONS	MIN	TYP†	MAX	UNIT
I_{CC}‡	Supply current	f = 14.4, MHz, V_{CC} = 5.5 V, T_A = 0°C to 70°C		28	65	mA

† All typical values are at T_A = 70°C and are used for thermal resistance calculations.
‡ I_{CC} characteristics are inversely proportional to temperature; i.e., I_{CC} decreases approximately linearly with temperature.

CLOCK CHARACTERISTICS AND TIMING

The TMS320C10-14 can use either its internal oscillator or an external frequency source for a clock.

internal clock option

The internal oscillator is enabled by connecting a crystal across X1 and X2/CLKIN (see Figure 1). The frequency of CLKOUT is one-fourth the crystal fundamental frequency. The crystal should be fundamental mode, and parallel resonant, with an effective series resistance of 30 ohms, a power dissipation of 1 mW, and be specified at a load capacitance of 20 pF.

PARAMETER	TEST CONDITIONS	MIN	NOM	MAX	UNIT
Crystal frequency, f_X	T_A = 0°C to 70°C	6.7		14.4	MHz
C1, C2	T_A = 0°C to 70°C		10		pF

external clock option

An external frequency source can be used by injecting the frequency directly into X2/CLKIN with X1 left unconnected. The external frequency injected must conform to the specifications listed in the table below.

switching characteristics over recommended operating conditions

		TEST CONDITIONS	MIN	NOM	MAX	UNIT
$t_{c(C)}$	CLKOUT cycle time§	R_L = 825 Ω, C_L = 100 pF, (see Figure 2)	277.78			ns
$t_{r(C)}$	CLKOUT rise time			10		ns
$t_{f(C)}$	CLKOUT fall time			8		ns
$t_{w(CL)}$	Pulse duration, CLKOUT low			131		ns
$t_{w(CH)}$	Pulse duration, CLKOUT high			129		ns
$t_{d(MCC)}$	Delay time, CLKIN↑ to CLKOUT↓		25¶		60¶	ns

§ $t_{c(C)}$ is the cycle time of CLKOUT, i.e., $4t_{c(MC)}$ (4 times CLKIN cycle time if an external oscillator is used).
¶ Values derived from characterization data and not tested.

timing requirements over recommended operating conditions

		MIN	NOM	MAX	UNIT
$t_{c(MC)}$	Master clock cycle time	69.5		150	ns
$t_{r(MC)}$	Rise time, master clock input		5¶	10¶	ns
$t_{f(MC)}$	Fall time, master clock input		5¶	10¶	ns
$t_{w(MCP)}$	Pulse duration, master clock	$0.4t_{c(MC)}$¶		$0.6t_{c(MC)}$¶	ns
$t_{w(MCL)}$	Pulse duration, master clock low, $t_{c(MC)}$ = 50 ns		20¶		ns
$t_{w(MCH)}$	Pulse duration, master clock high, $t_{c(MC)}$ = 50 ns		20¶		ns

¶ Values derived from characterization data and not tested.

MEMORY AND PERIPHERAL INTERFACE TIMING

switching characteristics over recommended operating conditions

PARAMETER		TEST CONDITIONS	MIN	NOM	MAX	UNIT
t_{d1}	Delay time, CLKOUT↓ to address bus valid	R_L = 825 Ω, C_L = 100 pF (see Figure 2)	10†		50	ns
t_{d2}	Delay time, CLKOUT↓ to $\overline{MEN}$↓		$1/4t_{c(C)} - 5$†		$1/4t_{c(C)} + 15$	ns
t_{d3}	Delay time, CLKOUT↓ to $\overline{MEN}$↑		−10†		15	ns
t_{d4}	Delay time, CLKOUT↓ to $\overline{DEN}$↓		$1/4t_{c(C)} - 5$†		$1/4t_{c(C)} + 15$	ns
t_{d5}	Delay time, CLKOUT↓ to $\overline{DEN}$↑		−10†		15	ns
t_{d6}	Delay time, CLKOUT↓ to $\overline{WE}$↓		$1/2t_{c(C)} - 5$†		$1/2t_{c(C)} + 15$	ns
t_{d7}	Delay time, CLKOUT↓ to $\overline{WE}$↑		−10†		15	ns
t_{d8}	Delay time, CLKOUT↓ to data bus OUT valid				$1/4t_{c(C)} + 65$	ns
t_{d9}	Time after CLKOUT↓ that data bus starts to be driven		$1/4t_{c(C)} - 5$†			ns
t_{d10}	Time after CLKOUT↓ that data bus stops being driven				$1/4t_{c(C)} + 40$†	ns
t_v	Data bus OUT valid after CLKOUT↓		$1/4t_{c(C)} - 10$			ns
$t_{h(A\text{-}WMD)}$	Address hold time after $\overline{WE}$↑, $\overline{MEN}$↑, or $\overline{DEN}$↑ (see Note 8)		−10†			ns
$t_{su(A\text{-}MD)}$	Address bus setup time prior to $\overline{MEN}$↓ or $\overline{DEN}$↓		$1/4t_{c(C)} - 45$			ns

† Values derived from characterization data and not tested.
NOTE 8: For interfacing I/O devices, see Figure 3.

timing requirements over recommended operating conditions

		TEST CONDITIONS	MIN	NOM	MAX	UNIT
$t_{su(D)}$	Setup time, data bus valid prior to CLKOUT↓	R_L = 825 Ω, C_L = 100 pF (see Figure 2)	50			ns
$t_{h(D)}$	Hold time, data bus held valid after CLKOUT↓ (see Note 9)		0			ns

NOTE 9: Data may be removed from the data bus upon $\overline{MEN}$↑ or $\overline{DEN}$↑ preceding CLKOUT↓.

RESET ($\overline{RS}$) TIMING

switching characteristics over recommended operating conditions

PARAMETER		TEST CONDITIONS	MIN	TYP	MAX	UNIT
t_{d11}	Delay time, $\overline{DEN}$↑, $\overline{WE}$↑, and $\overline{MEN}$↑ from $\overline{RS}$	R_L = 825 Ω, C_L = 100 pF (see Figure 2)			$1/2t_{c(C)}+50$†	ns
$t_{dis(R)}$	Data bus disable time after $\overline{RS}$				$1/4t_{c(C)}+50$†	ns

† Values were derived from characterization data and not tested.

timing requirements over recommended operating conditions

		MIN	NOM	MAX	UNIT
$t_{su(R)}$	Reset ($\overline{RS}$) setup time prior to CLKOUT (see Note 10)	50			ns
$t_{w(R)}$	$\overline{RS}$ pulse duration	$5t_{c(C)}$			ns

NOTE 10: $\overline{RS}$ can occur anytime during a clock cycle. Time given is minimum to ensure synchronous operation.

INTERRUPT ($\overline{INT}$) TIMING

timing requirements over recommended operating conditions

		MIN	NOM	MAX	UNIT
$t_{f(INT)}$	Fall time, $\overline{INT}$			15	ns
$t_{w(INT)}$	Pulse duration, $\overline{INT}$	$t_{c(C)}$			ns
$t_{su(INT)}$	Setup time, $\overline{INT}$↓ before CLKOUT↓	50			ns

IO ($\overline{BIO}$) TIMING

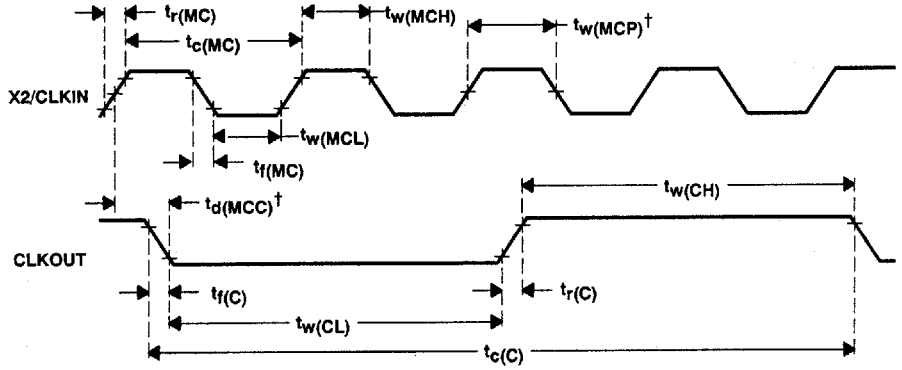
timing requirements over recommended operating conditions

		MIN	NOM	MAX	UNIT
$t_{f(IO)}$	Fall time, $\overline{BIO}$			15	ns
$t_{w(IO)}$	Pulse duration, $\overline{BIO}$	$t_{c(C)}$			ns
$t_{su(IO)}$	Setup time, $\overline{BIO}$↓ before CLKOUT↓	50			ns

TIMING DIAGRAMS

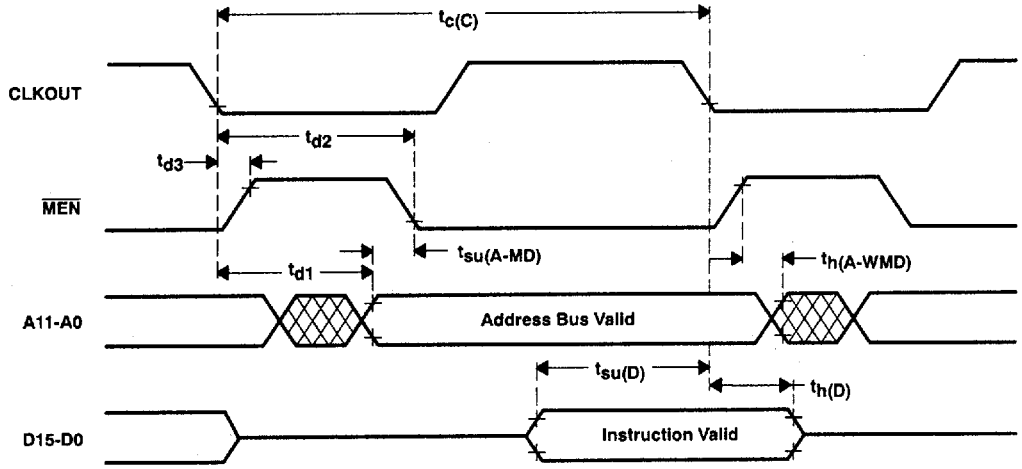
Timing measurements are referenced to and from a low voltage of 0.8 volts and a high voltage of 2 volts, unless otherwise noted.

clock timing

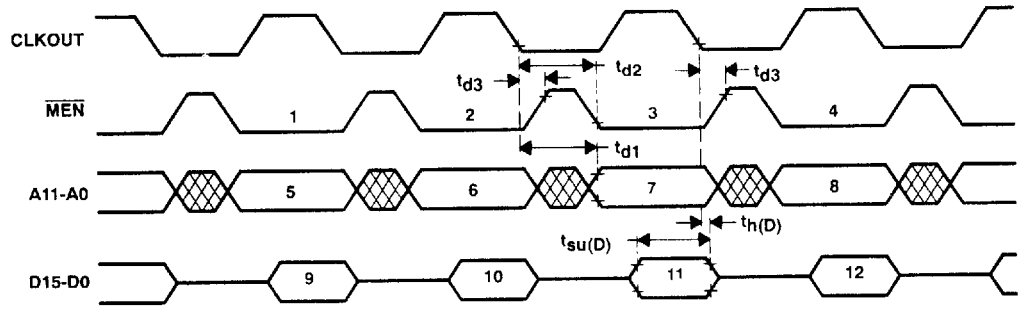


† $t_{d(MCC)}$ and $t_{w(MCP)}$ are referenced to an intermediate level of 1.5 V on the CLKIN waveform.

memory read timing



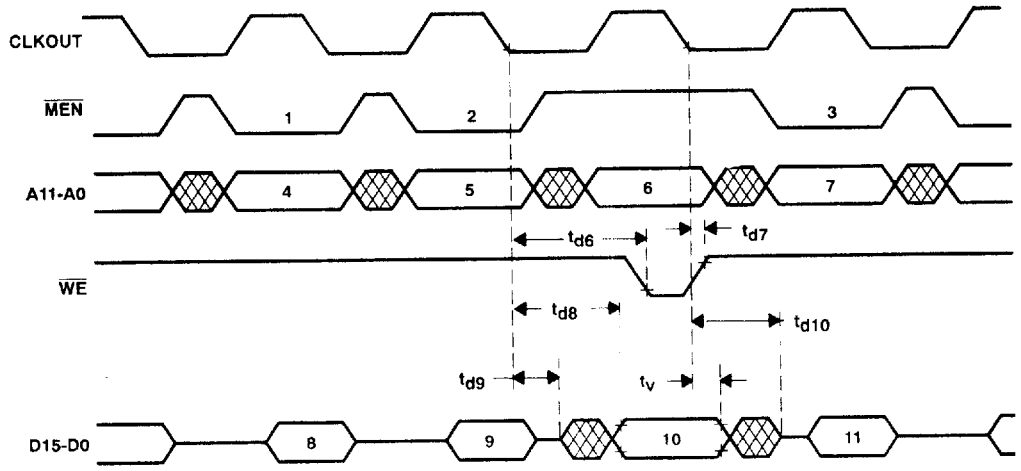
TBLR instruction timing



Legend:

1. TBLR Instruction Prefetch
2. Dummy Prefetch
3. Data Fetch
4. Next Instruction Prefetch
5. Address Bus Valid
6. Address Bus Valid
7. Address Bus Valid
8. Address Bus Valid
9. Instruction Valid
10. Instruction Valid
11. Data Input Valid
12. Instruction Valid

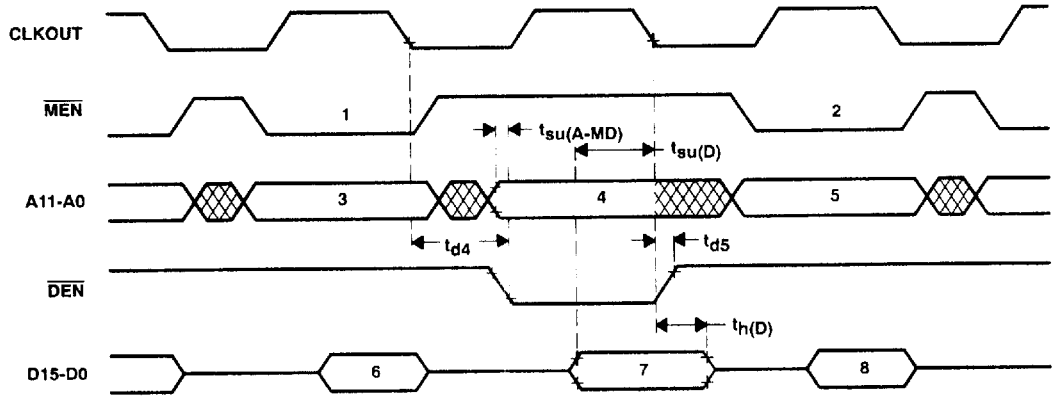
TBLW instruction timing



Legend:

1. TBLW Instruction Prefetch
2. Dummy Prefetch
3. Next Instruction Prefetch
4. Address Bus Valid
5. Address Bus Valid
6. Address Bus Valid
7. Address Bus Valid
8. Instruction Valid
9. Instruction Valid
10. Data Output Valid
11. Instruction Valid

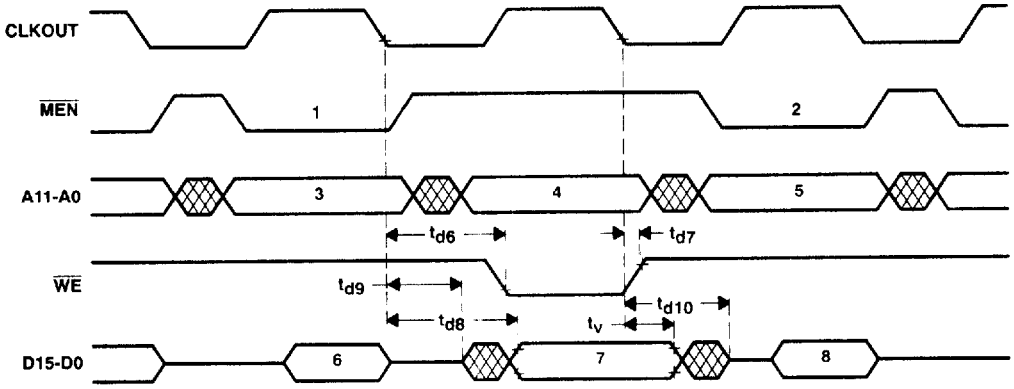
IN Instruction timing



Legend:

1. IN Instruction Prefetch
2. Next Instruction Prefetch
3. Address Bus Valid
4. Peripheral Address Valid
5. Address Bus Valic
6. Instruction Valid
7. Data Input Valid
8. Instruction Valid

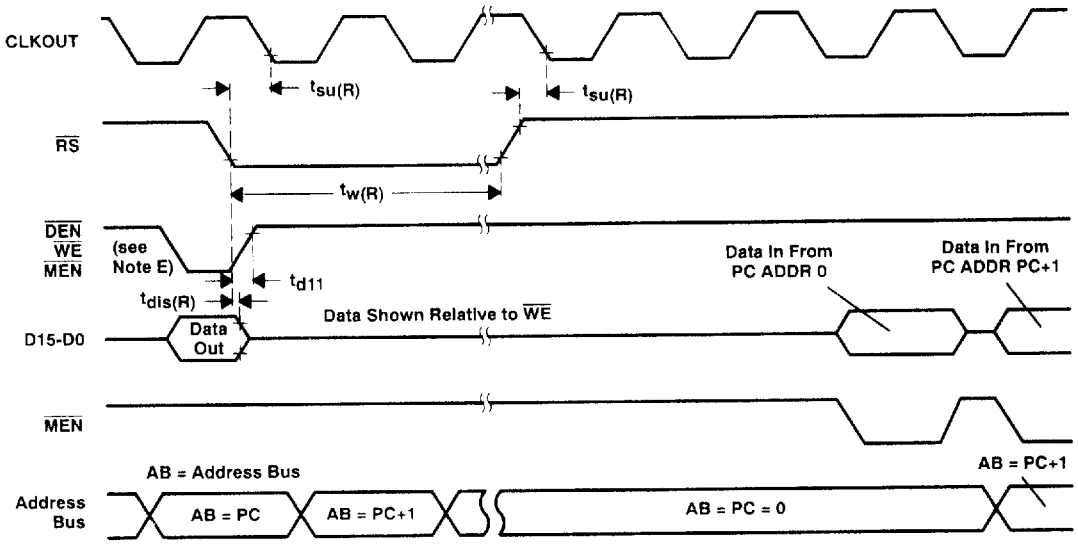
OUT Instruction timing



Legend:

1. OUT Instruction Prefetch
2. Next Instruction Prefetch
3. Address Bus Valid
4. Peripheral Address Valid
5. Address Bus Valid
6. Instruction Valid
7. Data Output Valid
8. Instruction Valid

reset timing



NOTES: A. $\overline{RS}$ forces $\overline{DEN}$, $\overline{WE}$, and $\overline{MEN}$ high and places data bus D0 through D15 in a high-impedance state. AB outputs (and program counter) are synchronously cleared to zero after the next complete CLK cycle from $\overline{RS}\downarrow$.

B. $\overline{RS}$ must be maintained for a minimum of five clock cycles.

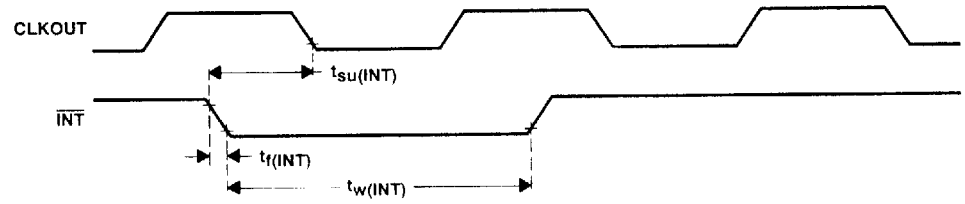
C. Resumption of normal program will commence after one complete CLK cycle from $\overline{RS}\uparrow$.

D. Due to the synchronization action on $\overline{RS}$, time to execute the function can vary dependent upon when $\overline{RS}\uparrow$ or $\overline{RS}\downarrow$ occur in the CLK cycle.

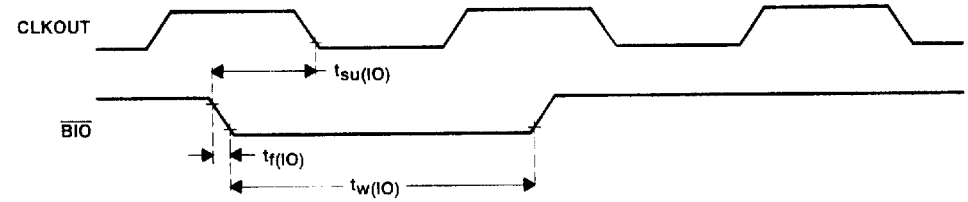
E. Diagram shown is for definition purpose only. $\overline{DEN}$, $\overline{WE}$, and $\overline{MEN}$ are mutually exclusive.

F. During a write cycle, $\overline{RS}$ may produce an invalid write address.

interrupt timing



$\overline{BIO}$ timing



TYPICAL POWER VS. FREQUENCY GRAPHS

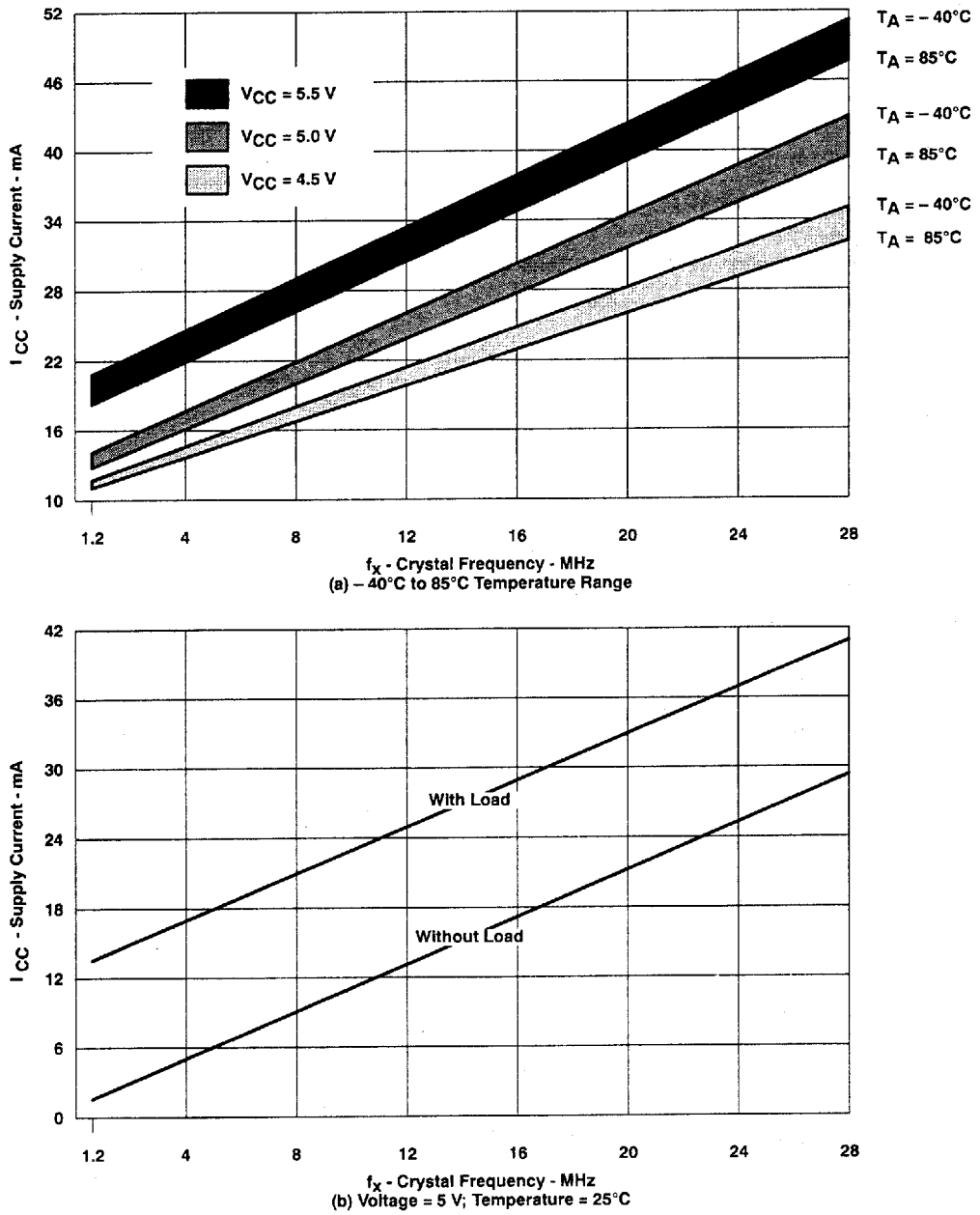
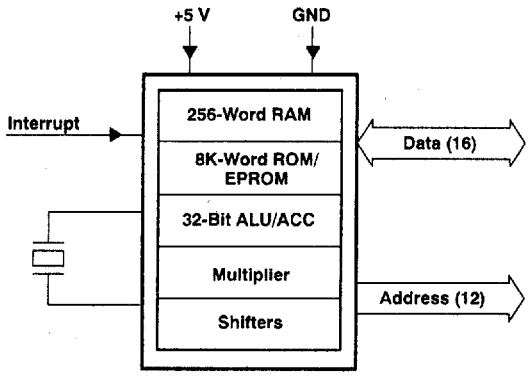


Figure 4. Typical CMOS I_{CC} vs Frequency

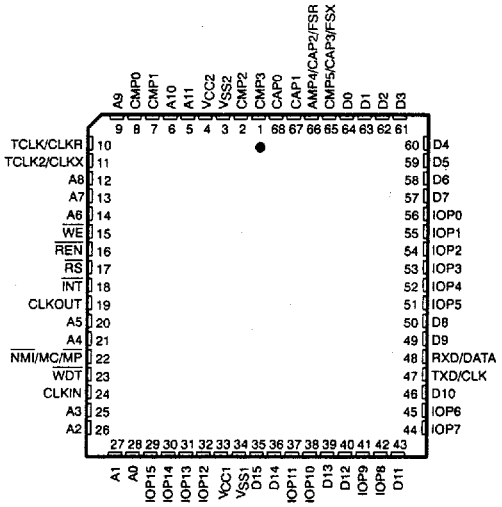
JANUARY 1987 — REVISED JULY 1991

Key Features: TMS320C14/E14/P14

- 160-ns Instruction Cycle
- 256 Words of On-Chip Data RAM
- 4K Words of On-Chip Program ROM (TMS320C14)
- 4K Words of On-Chip Program EPROM (TMS320E14/P14)
- One-Time Programmable (OTP) Windowless EPROM Version Available ('320P14)
- EPROM Code Protection for Copyright Security
- External Memory Expansion up to 4K-Words at Full Speed (Microprocessor Mode)
- 16 × 16-Bit Multiplier With 32-Bit Product
- 0 to 16-Bit Barrel Shifter
- Seven Input and Seven Output External Ports
- Bit Selectable I/O Port (16 Pins)
- 16-Bit Bidirectional Data Bus With Greater than 50-Mbps Transfer Rate
- Asynchronous Serial Port
- 15 Internal/External Interrupts
- Event Manager With Capture Inputs and Compare Outputs
- Four Independent Timers [Watchdog, General Purpose (2), Serial Port]
- Four-Level Hardware Stack
- Packaging: 68-Pin PLCC (FN Suffix) or CLCC (FZ Suffix)
- Single 5-V Supply
- Operating Free-Air Temperature . . . 0°C to 70°C



TMS320C14, TMS320E14/P14
FN/FZ Packages
(Top View)



Introduction

The 'C14/E14/P14 are 16/32-bit single-chip digital signal processing (DSP) microcontrollers that combine the high performance of a DSP with on-chip peripherals. With a 160-ns instruction cycle, these devices are capable of executing up to 6.4 million instructions per second (MIPS). The 'C14/E14/P14 DSPs are ideal for applications such as automotive control systems, computer peripherals, industrial controls, and military command/control system applications.

Control-specific on-chip peripherals include: An event manager with 6 channel PWM D/A/, 6-bit I/O pins, an asynchronous serial port, four 16-bit timers, and internal/external interrupts.

With 4K-words of on-chip ROM, the 'C14 is a mask programmable device. Code is provided by the customer, and TI incorporates the customer's code into the photomask. It is offered in a 68-pin plastic chip carrier package (FN suffix), rated for operation from 0°C to 70°C

The 'E14 is provided with a 4K-word on-chip EPROM. This EPROM version is excellent for prototyping and for customized applications. It is programmable with standard EPROM programmers. It is offered in a 68-pin (windowed) cerquad package (FZ suffix), rated for operation from 0°C to 70°C.

The 'P14 features a one-time programmable 4K-word on-chip EPROM. The 'P14 is provided in an unprogrammed state and is programmed as if it were a blank 'E14. It is offered in a low-cost, volume-production-oriented, 68-pin plastic leaded chip carrier (PLCC) package (FN suffix), rated for operation from 0°C to 70°C.

Each device can execute programs form either internal (MC/$\overline{MP}$=0) or external program memory (MC/$\overline{MP}$=1).

For proprietary code security, the 'E14 and 'P14 incorporate an EPROM protect bit (RBIT). If this bit is programmed, the device's internal program memory cannot be accessed by any external means.

TERMINAL FUNCTIONS

PIN NAME	PIN NO.	I/O/Z†	DESCRIPTION
			ADDRESS/DATA BUSES
A11	5	O/Z	Program memory address bus A11 (MSB) through A0 (LSB) and port addresses PA2 (MSB) through PA0 (LSB). Addresses A11 through A0 are always active and never go to high impedance except during reset. During execution of the IN and OUT instructions, pins 26, 27, and 28 carry the port addresses. Pins A3 through A11 are held high when port accesses are made on pins PA0 through PA2.
A10	6		
A9	9		
A8	12		
A7	13		
A6	14		
A5	20		
A4	21		
A3	25		
A2/PA2	26		
A1/PA1	27		
A0/PA0	28		
D15 MSB	35	I/O/Z	Parallel data bus D15 (MSB) through D0 (LSB). The data bus is always in the high-impedance state except when $\overline{WE}$ is active (low). The data bus is also active when internal peripherals are written to.
D14	36		
D13	39		
D12	40		
D11	43		
D10	46		
D9	49		
D8	50		
D7	57		
D6	58		
D5	59		
D4	60		
D3	61		
D2	62		
D1	63		
D0 LSB	64		
			INTERRUPT AND MISCELLANEOUS SIGNALS
$\overline{INT}$	18	I	External interrupt input. The interrupt signal is generated by a high-to-low transition on this pin.
$\overline{NMI}$/MC/$\overline{MP}$	22	I	Non-maskable interrupt. When this pin is brought low, the device is interrupted irrespective of the state of the INTM bit in status register ST. Microcomputer/microprocessor select. This pin is also sampled when RS is low. If high during reset, internal program memory is selected. If low during reset, external memory will be selected.
$\overline{WE}$	15	O	Write enable. When active low, $\overline{WE}$ indicates that device will output data on the bus.
$\overline{REN}$	16	O	Read enable. When active low, $\overline{REN}$ indicates that device will accept data from the bus.
$\overline{RS}$	17	I	Reset. When this pin is low, the device is reset and PC is set to zero.

Continued next page.

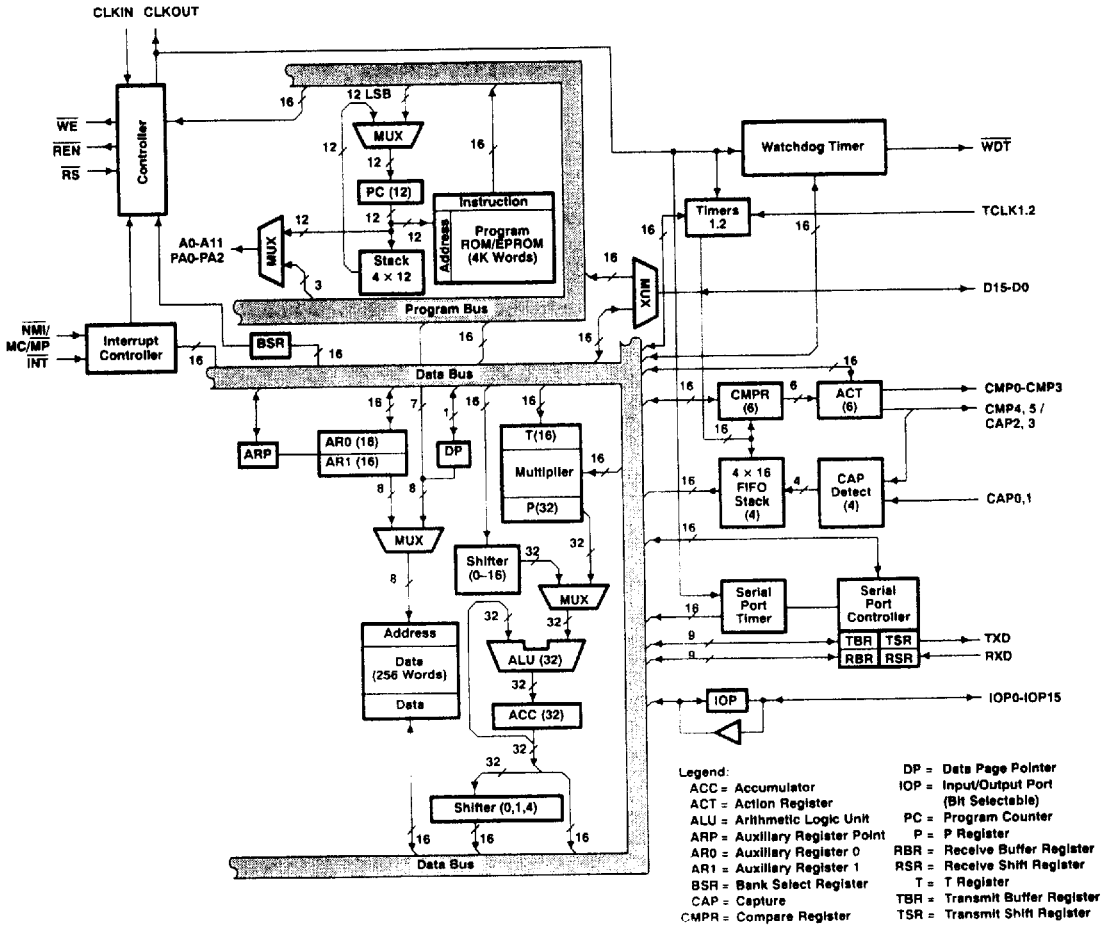
† Input/Output/High-impedance state.

TERMINAL FUNCTIONS (concluded)

PIN NAME	PIN NO.	I/O/Z†	DESCRIPTION
			SUPPLY/OSCILLATOR SIGNALS
CLKOUT	19	O	System clock output (one fourth CLKIN frequency).
V_{CC}	4,33	I	5-V supply pins.
V_{SS}	3,34	I	Ground pins.
CLKIN	24	I	Master clock input from external clock source.
			SERIAL PORT AND TIMER SIGNALS
RXD	48	I	Asynchronous mode receive input.
TXD	47	O/Z	Asynchronous mode transmit output.
TCLK1	10	I	Timer 1 clock. If external clock is selected, it serves as clock input to Timer 1.
TCLK2	11	I	Timer 2 clock. If external clock is selected, it serves as clock input to Timer 2.
$\overline{WDT}$	23	O	Watchdog timer output. An active low is generated on this pin when the watchdog timer times out.
			BIT I/O PINS
IOP15 MSB	29	I/O	16 bit I/O lines that can be individually configured as inputs or outputs and also individually set or reset when configured as outputs.
IOP14	30		
IOP13	31		
IOP12	32		
IOP11	37		
IOP10	38		
IOP9	41		
IOP8	42		
IOP7	44		
IOP6	45		
IOP5	51		
IOP4	52		
IOP3	53		
IOP2	54		
IOP1	55		
IOP0 LSB	56		
			COMPARE AND CAPTURE SIGNALS
CMP0	8	O	Compare outputs. The states of these pins are determined by the combination of compare and action registers.
CMP1	7		
CMP2	2		
CMP3	1		
CAP0	68	I	Capture inputs. A transition on these pins causes the timer register to be captured in FIFO stack.
CAP1	67		
CMP4/CAP2	66	I/O	This pin can be configured as compare output or capture input.
CMP5/CAP3	65	I/O	This pin can be configured as compare output or capture input.

† Input/Output/High-impedance state.

functional block diagram



architecture

The 'C1x family utilizes a modified Harvard architecture for speed and flexibility. In a strict Harvard architecture, program and data memory lie in two separate spaces, permitting a full overlap of instruction fetch and execution. The 'C1x family's modification of a Harvard architecture allows transfers between program and data spaces, thereby increasing the flexibility of the device. This modification permits coefficients stored in program memory to be read into the RAM, eliminating the need for a separate coefficient ROM. It also makes available immediate instructions and subroutines based on computed values.

32-bit ALU/accumulator

The 'C14/E14/P14 devices contain a 32-bit ALU and accumulator for support of double-precision, twos-complement arithmetic. The ALU is a general-purpose arithmetic unit that operates on 16-bit words taken from the data RAM or derived from immediate instructions. In addition to the usual arithmetic instructions, the ALU can perform Boolean operations, providing the bit manipulation ability required of a high-speed controller.

The accumulator stores the output from the ALU and is often an input to the ALU. It operates with a 32-bit wordlength. The accumulator is divided into a high-order word (bits 31 through 16) and a low-order word (bits 15 through 0). Instructions are provided for storing the high- and low- order accumulator words in memory.

shifters

Two shifters are available for manipulating data. The ALU barrel shifter performs a left-shift of 0 to16 places on data memory words loaded into the ALU. This shifter extends the high-order bit of the data word and zero-fills the low-order bits for twos-complement arithmetic. The accumulator parallel shifter performs a left-shift of 0, 1, or 4 places on the entire accumulator and places the resulting high-order accumulator bits into data RAM. Both shifters are useful for scaling and bit extraction

16 × 16-bit parallel multiplier

The multiplier performs a 16 × 16-bit twos-complement multiplication with a 32-bit result in a single instruction cycle. The multiplier consists of three units: the T Register, P Register, and the multiplier array. The 16-bit T Register temporarily stores the multiplicand; the P Register stores the 32-bit product. Multiplier values either come from the data memory or are derived immediately from the MPYK (multiply immediate) instruction word. The fast on-chip multiplier allows the device to perform fundamental operations such as convolution, correlation, and filtering.

data and program memory

Since the 'C14/E14/P14 devices use a Harvard architecture, data and program memory reside in two separate spaces. These devices have 256 words of on-chip data RAM and 4K words of on-chip program ROM ('C14) or EPROM ('E14 and the OTP 'P14). The EPROM cell utilizes standard PROM programmers and is programmed identically to a 64K-bit CMOS EPROM (TMS27C64)

program memory expansion

The 'C1x devices are capable of executing up to 4K words of external memory at full speed for those applications requiring external program memory space. This allows for external RAM-based systems to provide multiple functionality.

microcomputer/microprocessor operating modes

The 'C14/E14/P14 devices offer two modes of operation defined by the state of the $\overline{\text{NMI}}$/MC/$\overline{\text{MP}}$ pin during reset: the microcomputer mode ($\overline{\text{NMI}}$/MC/$\overline{\text{MP}}$ is high) or the microprocessor mode ($\overline{\text{NMI}}$/MC/$\overline{\text{MP}}$ is low). In the microcomputer mode, the on-chip ROM is mapped into the program memory space. In the microprocessor mode, all 4K words of memory are external.

interrupts and subroutines

The 'C14/E14/P14 devices contain a four-level hardware stack for saving the contents of the program counter during interrupts and subroutine calls. Instructions are available for saving the complete context of the device. PUSH and POP instructions permit a level of nesting restricted only by the amount of available RAM. The 'C14/E14/P14 have a total of 15 internal/external interrupts. Fourteen of these are maskable; $\overline{\text{NMI}}$ is the fifteenth.

input/output

The 16-bit parallel data bus can be utilized to access external peripherals. However, only the lower three address lines are active. The upper nine address lines are driven high.

bit I/O

The 'C14/E14/P14 has 16 pins of bit I/O that can be individually configured as inputs or outputs. Each of the pins can be set or cleared without affecting the others. The input pins can also detect and match patterns and generate a maskable interrupt signal to the CPU.

serial port

The 'C14/E14/P14 includes an I/O-mapped asynchronous serial port.

event manager

An event manager is included that provides up to four capture inputs and up to six compare outputs. This peripheral operates with the timers to provide a form of programmable event logging/detection. The six compare outputs can also be configured to produce six channels of high precision PWM.

timers 1 and 2

Two identical 16-bit timers are provided for general purpose applications. Both timers include a 16-bit period register and buffer latch, and can generate a maskable interrupt.

serial port timer

The serial port timer is a 16-bit timer primarily intended for baud rate generation for the serial port. Its architecture is the same as timers 1 and 2, therefore it can serve as a general purpose timer if not needed for serial communication.

watchdog timer

The 'C14/E14/P14 contain a 16-bit watchdog timer that can produce a timeout ($\overline{WDT}$) signal for various applications such as software development and event monitoring. The watchdog timer also generates, at the point of the timeout, a maskable interrupt signal to the CPU.

Instruction set

A comprehensive instruction set supports both numeric-intensive operations, such as signal processing, and general-purpose operations, such as high-speed control. All of the first-generation devices are object-code compatible and use the same 60 instructions. The instruction set consists primarily of single-cycle single-word instructions, permitting execution rates of more than six million instructions per second. Only infrequently used branch and I/O instructions are multicycle. Instructions that shift data as part of an arithmetic operation execute in a single cycle and are useful for scaling data in parallel with other operations.

NOTE

The $\overline{BIO}$ pin on other 'C1x devices is not available for use in the 'C14/E14/P14 devices. An attempt to execute the BIOZ (Branch on $\overline{BIO}$ low) instruction will result in a two cycle NOP action.

Three main addressing modes are available with the instruction set: direct, indirect, and immediate addressing.

direct addressing

In direct addressing, seven bits of the instruction word concatenated with the 1-bit data page pointer from the data memory address. This implements a paging scheme in which each page contains 128 words.

indirect addressing

Indirect addressing forms the data memory address from the least-significant eight bits of one of the two auxiliary registers, AR0 and AR1. The Auxiliary Register Pointer (ARP) selects the current auxiliary register. The auxiliary registers can be automatically incremented or decremented and the ARP changed in parallel with the execution of any indirect instruction to permit single-cycle manipulation of data tables. Indirect addressing can be used with all instructions requiring data operands, except for the immediate operand instructions.

immediate addressing

Immediate instructions derive data from part of the instruction word rather than from part of the data RAM. Some useful immediate instructions are multiply immediate (MPYK), load accumulator immediate (LACK), and load auxiliary register immediate (LARK).

electrical specifications

This section contains all the electrical specifications for the 'C14/E14/P14 devices, including test parameter measurement information. Parameters with $_{PP}$ subscripts apply only to the 'E14 and 'P14 in the EPROM programming mode.

absolute maximum ratings over specified temperature range (unless otherwise noted)†

Supply voltage range, V_{CC} (see Note 6) ..... – 0.3 V to 7 V
Supply voltage range, V_{PP} (see Note 6) ..... – 0.6 V to 14 V
Input voltage range ..... – 0.3 V to 14 V
Output voltage range ..... – 0.3 V to 7 V
Continuous power dissipation ..... 0.5 W
Air temperature range above operating device: L version ..... 0 °C to 70 °C
Storage temperature ..... – 55 °C + 150 °C

† Stresses beyond those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. This is a stress rating only, and functional operation of the device at these or any other conditions beyond those indicated in the "Recommended Operating Conditions" section of this specification is not implied. Exposure to absolute-maximum-rated conditions for extended periods may affect device reliability.

NOTE 6: All voltage values are with respect to V_{SS}.

recommended operating conditions

			MIN	NOM	MAX	UNIT
V_{CC}	Supply voltage	Operating voltage	4.75	5	5.25	V
		Fast programming	5.75	6	6.25	V
		SNAP! Pulse programming	6.25	6.5	6.75	V
V_{PP}	Supply voltage for Fast programming (see Note 11)		12.25	12.5	12.75	V
V_{PP}	Supply voltage for SNAP! Pulse programming (see Note 11)		12.75	13	13.25	V
V_{SS}	Supply voltage			0		V
V_{IH}	High-level input voltage	CLKIN, CAP0, CAP1, CMP4/CAP2, CMP5/CAP3, $\overline{RS}$	3			V
		All remaining inputs	2			
V_{IL}	Low-level input voltage, all inputs				0.8	V
I_{OH}	High-level output current, all outputs				– 300	μA
I_{OL}	Low-level output current, all outputs				2	mA
T_A	Operating free-air temperature		0		70	°C

NOTE 11: V_{PP} can be applied only to programming pins designed to accept V_{PP} as an input. During programming the total supply current is $I_{PP} + I_{CC}$.

electrical characteristics over specified temperature range (unless otherwise noted)

PARAMETER			TEST CONDITIONS		MIN	TYP†	MAX	UNIT
V_{OH}	High-level output voltage		I_{OH} = MAX		2.4	3		V
			I_{OH} = 20 μA (see Note 7)		V_{CC} – 0.4†			V
V_{OL}	Low-level output voltage		I_{OL} = MAX			0.3	0.5	V
I_{OZ}	Off-state output voltage		V_{CC} = MAX	V_O = 2.4 V			20	μA
				V_O = 0.4 V			– 20	
I_I	Input current		V_I = V_{SS} to V_{CC}	All other inputs except CLKIN			± 20	μA
				CLKIN			± 50	
I_{CC}§	Supply current		f = 25 6 MHz, V_{CC} = 5.25 V, T_A = 0°C to 70°C			70	90	mA
I_{PP1}	V_{PP} supply current		V_{PP} = V_{CC} = 5.5 V				100	μA
I_{PP2}	V_{PP} supply current (during program pulse)		V_{PP} = 13 V			30	50	mA
C_I	Input capacitance	Data bus	f = 1 MHz, All other pins 0 V			25‡		pF
		All others				15‡		
C_O	Output capacitance	Data bus				25‡		pF
		All others				10‡		

† All typical values are at V_{CC} = 5 V, T_A = 25°C, except I_{CC} at 70°C.
‡ Values derived from characterization data and not tested.
§ I_{CC} characteristics are inversely proportional to temperature.
NOTE 7: This voltage specification is included for interface to HC logic. However, note that all of the other timing parameters defined in this data sheet are specified for TTL logic levels and will differ for HC logic levels

PARAMETER MEASUREMENT INFORMATION

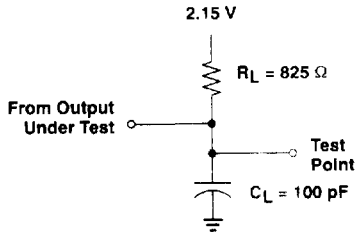


Figure 5. Test Load Circuit

EXTERNAL CLOCK REQUIREMENTS

The TMS320C14/E14/P14 use an external frequency source for a clock. This source is applied to the CLKIN pin, and must conform to the specifications in the table below.

PARAMETERS		TEST CONDITIONS	MIN	NOM	MAX	UNIT
CLKIN	Input clock frequency	T_A = 0°C to 70°C	6.7		25.6	MHz

CLOCK TIMING

switching characteristics over recommended operating conditions

PARAMETER		TEST CONDITIONS	MIN	NOM	MAX	UNIT
$t_{c(C)}$	CLKOUT cycle time ‡	R_L = 825 Ω, C_L = 100 pF, (see Figure 2)	156.25		600	ns
$t_{r(C)}$	CLKOUT rise time			10†		ns
$t_{f(C)}$	CLKOUT fall time			8†		ns
$t_{w(CL)}$	Pulse duration, CLKOUT low			72†		ns
$t_{w(CH)}$	Pulse duration, CLKOUT high			70†		ns
$t_{d(MCC)}$	Delay time CLKIN↑ to CLKOUT↓			45†		ns

† Values were derived from characterization data and not tested.

timing requirements over recommended operating conditions

		MIN	NOM	MAX	UNIT
$t_{c(MC)}$	Master clock cycle time ‡	39.06	40	150	ns
$t_{r(MC)}$	Rise time, master clock input		5†	10†	ns
$t_{f(MC)}$	Fall time, master clock input		5†	10†	ns
$t_{w(MCP)}$	Pulse duration, master clock	0.45 $t_{c(MC)}$†		0.55 $t_{c(MC)}$†	ns
$t_{w(MCL)}$	Pulse duration, master clock low		15†	130	ns
$t_{w(MCH)}$	Pulse duration, master clock high		15†	130	ns

† Values were derived from characterization data and not tested.
‡ $t_{c(C)}$ is the cycle time of CLKOUT, i.e., $4t_{c(MC)}$ (4 times CLKIN cycle time if an external oscillator is used).

MEMORY READ AND INSTRUCTION TIMING

switching characteristics over recommended operating conditions

PARAMETER		TEST CONDITIONS	MIN	NOM	MAX	UNIT
$t_{su(A)R}$	Address bus valid before $\overline{REN}\downarrow$	R_L = 825 Ω, C_L = 100 pF, (see Figure 2)	0.25 $t_{c(C)}$ – 39			ns
$t_{su(A)W}$	Address bus valid before $\overline{WE}\downarrow$		0.50 $t_{c(C)}$ – 45			ns
$t_{h(A)}$	Address bus valid after $\overline{REN}\uparrow$ or $\overline{WE}\uparrow$		5†			ns
$t_{en(D)W}$	Data starts being driven before $\overline{WE}\downarrow$				0.25 $t_{c(C)}$†	ns
$t_{su(D)W}$	Data valid prior to $\overline{WE}\downarrow$		0.25 $t_{c(C)}$ – 45			ns
$t_{h(D)W}$	Data valid after $\overline{WE}\uparrow$		0.25 $t_{c(C)}$ – 10			ns
$t_{dis(D)W}$	Data in high impedance after $\overline{WE}\uparrow$				0.25 $t_{c(C)}$ + 25†	ns
$t_{w(WEL)}$	$\overline{WE}$-low duration		0.50 $t_{c(C)}$ – 15			ns
$t_{w(RENL)}$	$\overline{REN}$-low duration		0.75 $t_{c(C)}$ – 15			ns
$t_{rec(WE)}$	Write recovery time, time between $\overline{WE}\uparrow$ and $\overline{REN}\downarrow$		0.25 $t_{c(C)}$ – 5			ns
$t_{rec(REN)}$	Read recovery time, time between $\overline{REN}\uparrow$ and $\overline{WE}\downarrow$		0.50 $t_{c(C)}$ – 10			ns
$t_{d(WE-CLK)}$	Time from $\overline{WE}\uparrow$ to CLKOUT↑		0.50 $t_{c(C)}$ – 15			ns

† Values were derived from characterization data and not tested.

timing requirements over recommended operating conditions

		TEST CONDITIONS	MIN	NOM	MAX	UNIT
$t_{su(D)R}$	Data set-up prior to $\overline{REN}\uparrow$	R_L = 825 Ω, C_L = 100 pF, (see Figure 2)	52			ns
$t_{h(D)R}$	Data hold after $\overline{REN}\uparrow$		0			ns
$t_{a(A)}$	Access time for read cycle data valid after valid address				$t_{c(C)}$ – 90	ns
$t_{oe(REN)}$	Access time for read cycle from $\overline{REN}\downarrow$				0.75 $t_{c(C)}$ – 60	ns
$t_{dis(D)R}$	Data in high impedance after $\overline{REN}\uparrow$				0.25 $t_{c(C)}$†	ns

RESET ($\overline{RS}$) TIMING

switching characteristics over recommended operating conditions

PARAMETER		TEST CONDITIONS	MIN	NOM	MAX	UNIT
$t_{d(RS-RW)}$	Delay from $\overline{RS}\downarrow$ to $\overline{REN}\uparrow$ and $\overline{WE}\uparrow$	R_L = 825 Ω, C_L = 100 pF, (see Figure 2)			0.75 $t_{c(C)}$ + 20†	ns
$t_{dis(RS-RW)}$	Delay from $\overline{RS}\downarrow$ to $\overline{REN}$ and $\overline{WE}$ into high impedance				1.25 $t_{c(C)}$†	ns
$t_{dis(RS-DB)}$	Data bus disable after $\overline{RS}\downarrow$				1.25 $t_{c(C)}$†	ns
$t_{dis(RS-AB)}$	Address bus disable after $\overline{RS}\downarrow$				$t_{c(C)}$†	ns
$t_{en(RS-AB)}$	Address bus enable after $\overline{RS}\uparrow$				$t_{c(C)}$†	ns

timing requirements over recommended operating conditions

		TEST CONDITIONS	MIN	NOM	MAX	UNIT
$t_{su(RS)}$	$\overline{RS}$ setup prior to CLKOUT↓ (see Note 10)	R_L = 825 Ω, C_L = 100 pF, (see Figure 2)	60			ns
$t_{w(RS)}$	$\overline{RS}$ pulse duration		5$t_{c(C)}$			ns

NOTE 10: $\overline{RS}$ can occur anytime during the clock cycle. Time given is minimum to ensure synchronous operation.

MICROCOMPUTER/MICROPROCESSOR MODE ($\overline{NMI}$/MC/$\overline{MP}$)

timing requirements over recommended operating conditions

		MIN	NOM	MAX	UNIT
$t_{h(MC/MP)}$‡	Hold time after $\overline{RS}$ high	$t_{c(C)}$			ns

† Values were derived from characterization data and not tested.
‡ Hold time to put device in microprocessor mode.

INTERRUPT ($\overline{INT}$)/NONMASKABLE INTERRUPT ($\overline{NMI}$)

timing requirements over recommended operating conditions

		MIN	NOM	MAX	UNIT
$t_{f(INT)}$	Fall time, $\overline{INT}$			15†	ns
$t_{f(NMI)}$	Fall time, $\overline{NMI}$			15†	ns
$t_{w(INT)}$	Pulse duration, $\overline{INT}$	$t_{c(C)}$			ns
$t_{w(NMI)}$	Pulse duration, $\overline{NMI}$	$t_{c(C)}$			ns
$t_{su(INT)}$	Setup time, $\overline{INT}$ before CLKOUT low (see Note 12)	60			ns
$t_{su(NMI)}$	Setup time, $\overline{NMI}$ before CLKOUT low (see Note 12)	60			ns

NOTE 12: $\overline{INT}$ and $\overline{NMI}$ are synchronous inputs and can occur at any time during the cycle. $\overline{NMI}$ and $\overline{INT}$ are edge triggered only.

BIT I/O TIMING

switching characteristics over recommended operating conditions

PARAMETER		TEST CONDITIONS	MIN	NOM	MAX	UNIT
$t_{rfo(IOP)}$	Rise and fall time outputs	R_L = 825 Ω, C_L = 100 pF, (see Figure 2)			20†	ns
$t_{d(IOP)}$	CLKOUT low to data valid outputs				0.75 $t_{c(C)}$ + 80	ns

timing requirements over recommended operating conditions

		TEST CONDITIONS	MIN	NOM	MAX	UNIT
$t_{rfi(IOP)}$	Rise and fall time inputs	R_L = 825 Ω, C_L = 100 pF, (see Figure 2)			20†	ns
$t_{su(IOP)}$	Data setup time before CLKOUT time		40			ns
$t_{w(IOP)}$	Input pulse duration		$t_{c(C)}$			ns

GENERAL PURPOSE TIMERS

timing requirements over recommended operating conditions

		TEST CONDITIONS	MIN	NOM	MAX	UNIT
$t_{r(TIM)}$	TCLK1, TCLK2 rise time	R_L = 825 Ω, C_L = 100 pF, (see Figure 2)			20†	ns
$t_{f(TIM)}$	TCLK1, TCLK2 fall time				20†	ns
$t_{wl(TIM)}$	TCLK1, TCLK2 low time		$t_{c(C)}$ + 20			ns
$t_{wh(TIM)}$	TCLK1, TCLK2 high time		$t_{c(C)}$ + 20			ns
$t_{clk(TIM)}$	Input pulse duration		2 $t_{c(C)}$ + 40			ns

† Values were derived from characterization data and not tested.

WATCHDOG TIMER TIMING

switching characteristics over recommended operating conditions

PARAMETER		TEST CONDITIONS	MIN	NOM	MAX	UNIT
$t_{f(WDT)}$	Fall time, $\overline{WDT}$	R_L = 825 Ω, C_L = 100 pF, (see Figure 2)			20†	ns
$t_{d(WDT)}$	CLKOUT to $\overline{WDT}$ valid		0.25 $t_{c(C)}$ + 20			ns
$t_{w(WDT)}$	WDT output pulse duration		7 $t_{c(C)}$			ns

EVENT MANAGER TIMER

switching characteristics over recommended operating conditions

PARAMETER		TEST CONDITIONS	MIN	NOM	MAX	UNIT
$t_{f(CMP)}$	Fall time, CMP0-CMP5	R_L = 825 Ω, C_L = 100 pF, (see Figure 2)			20†	ns
$t_{r(CMP)}$	Rise time, CMP0-CMP5				20†	ns

timing requirements over recommended operating conditions

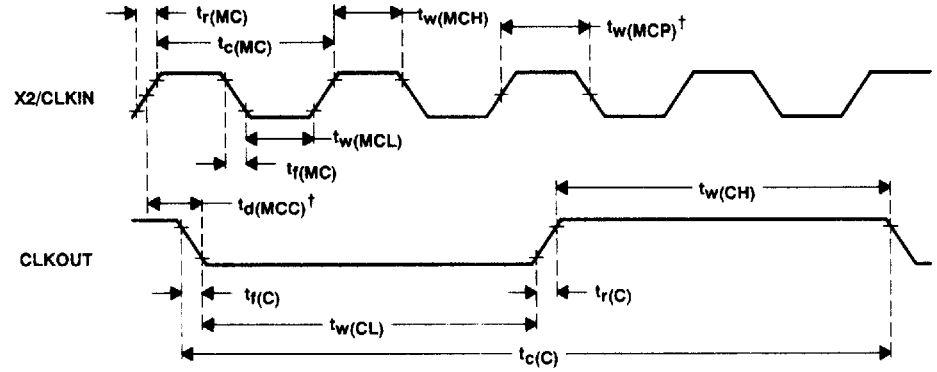
		TEST CONDITIONS	MIN	NOM	MAX	UNIT
$t_{w(CAP)}$	CAP0-CAP3 input pulse duration	R_L = 825 Ω, C_L = 100 pF, (see Figure 2)	$t_{c(C)}$ + 20			ns
$t_{su(CAP)}$	Capture input setup time before CLKOUT low		20†			ns

† Values were derived from characterization data and not tested.

TIMING DIAGRAMS

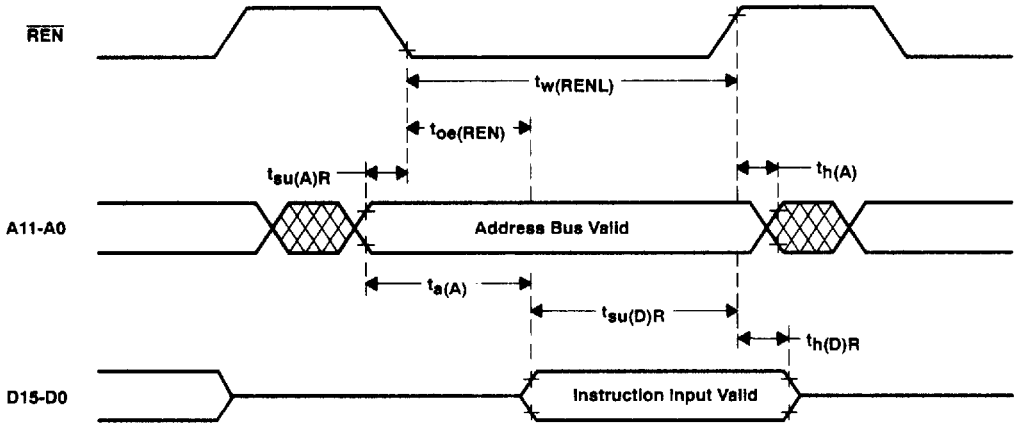
Timing measurements are referenced to and from a low voltage of 0.8 volts and a high voltage of 2 volts, unless otherwise noted.

clock timing

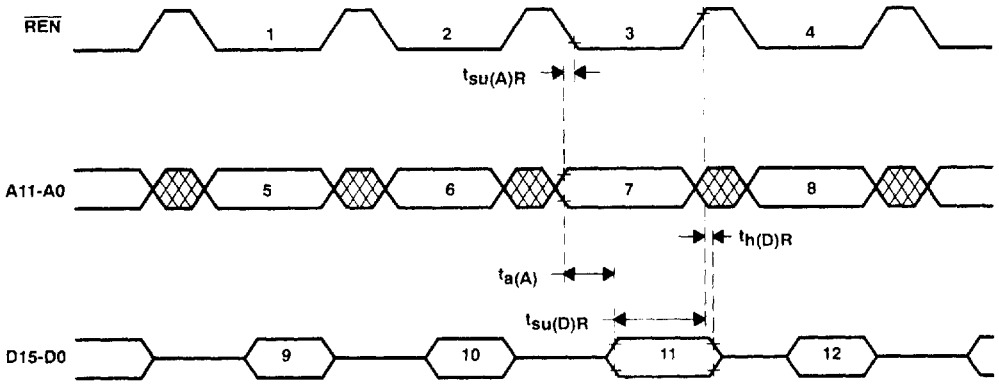


† $t_{d(MCC)}$ and $t_{w(MCP)}$ are referenced to an intermediate level of 1.5 V on the CLKIN waveform.

memory read timing



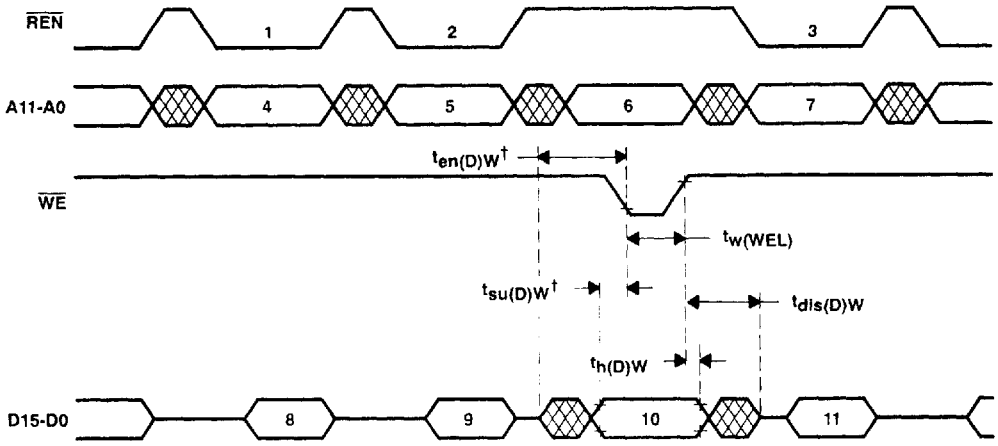
TBLR instruction timing



Legend:

1. TBLR Instruction Prefetch
2. Dummy Prefetch
3. Data Fetch
4. Next Instruction Prefetch
5. Address Bus Valid
6. Address Bus Valid
7. Address Bus Valid
8. Address Bus Valid
9. Instruction Input Valid
10. Instruction Input Valid
11. Data Input Valid
12. Instruction Input Valid

TBLW instruction timing

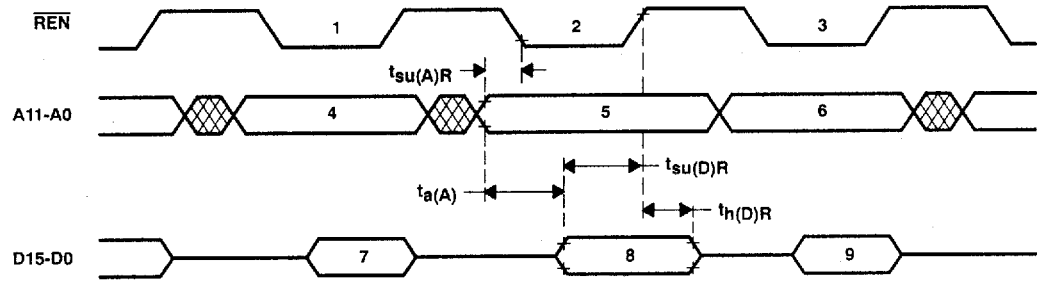


† Data valid prior to $\overline{WE}\downarrow$

Legend:

1. TBLW Instruction Prefetch
2. Dummy Prefetch
3. Next Instruction Prefetch
4. Address Bus Valid
5. Address Bus Valid
6. Address Bus Valid
7. Address Bus Valid
8. Instruction Input Valid
9. Instruction Input Valid
10. Data Output Valid
11. Instruction Input Valid

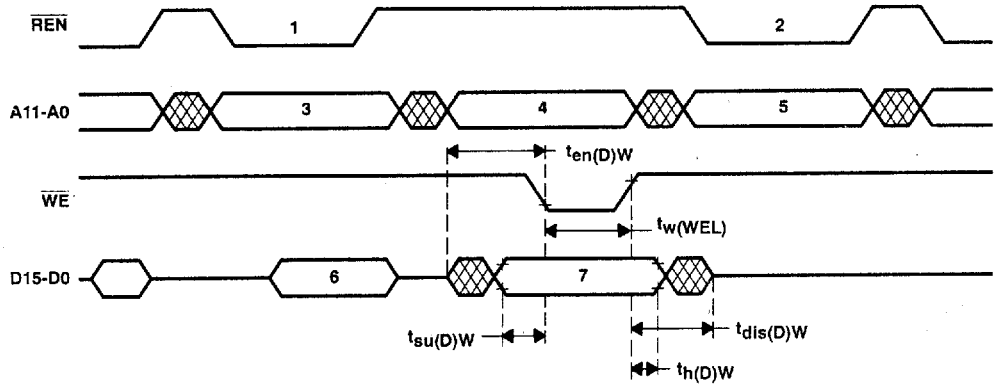
IN instruction timing



Legend:

1. IN Instruction Prefetch
2. Data Fetch
3. Next Instruction Prefetch
4. Address Bus Valid
5. Peripheral Address Valid
6. Address Bus Valid
7. Instruction Input Valid
8. Data Input Vaiid
9. Instruction Input Valid

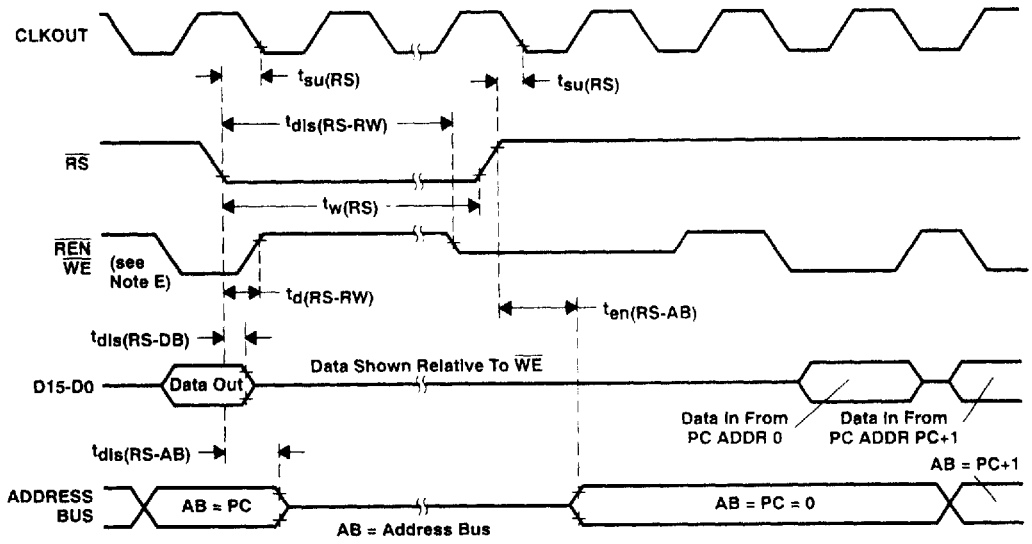
OUT instruction timing



Legend:

1. OUT Instruction Prefetch
2. Next Instruction Prefetch
3. Address Bus Valid
4. Peripheral Address Valid
5. Address Bus Valid
6. Instruction Input Valid
7. Data Output Valid

reset timing



NOTES: A. $\overline{RS}$ forces $\overline{REN}$, and $\overline{WE}$ high and then places data bus D0-D15, $\overline{REN}$, $\overline{WE}$, and address bus A0-A11 in a high-impedance state. AB outputs (and program counter) are synchronously cleared to zero after the next complete CLK cycle from $\overline{RS}\uparrow$.

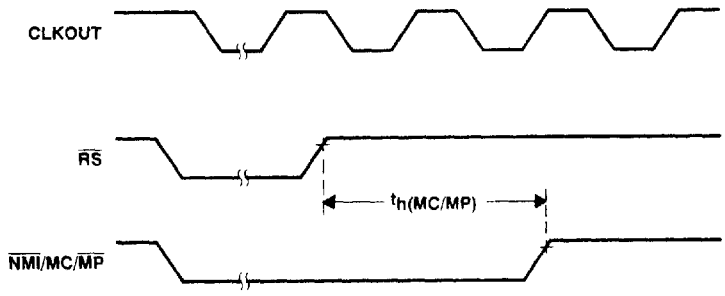
B. $\overline{RS}$ must be maintained for a minimum of five clock cycles.

C. Resumption of normal program will commence after one complete CLK cycle from $\overline{RS}\uparrow$.

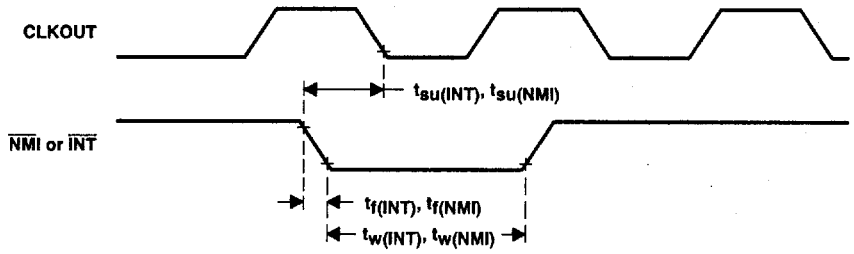
D. Due to the synchronization action on $\overline{RS}$, time to execute the function can vary dependent upon when $\overline{RS}\uparrow$ or $\overline{RS}\downarrow$ occur in the CLK cycle.

E. Diagram shown is for definition purpose only. $\overline{WE}$ and $\overline{REN}$ are mutually exclusive.

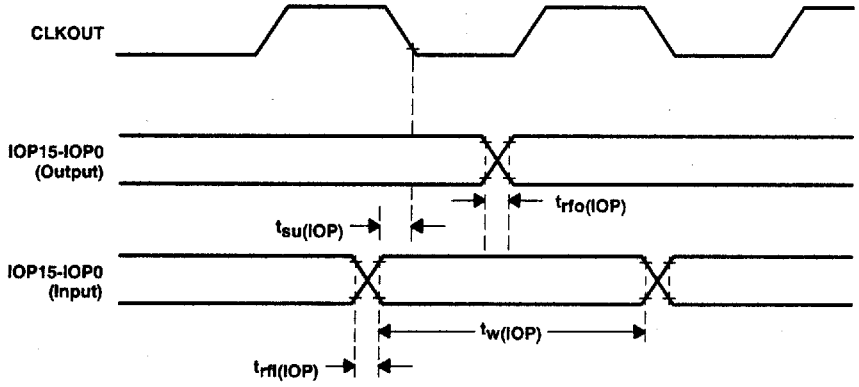
microcomputer/microprocessor mode timing



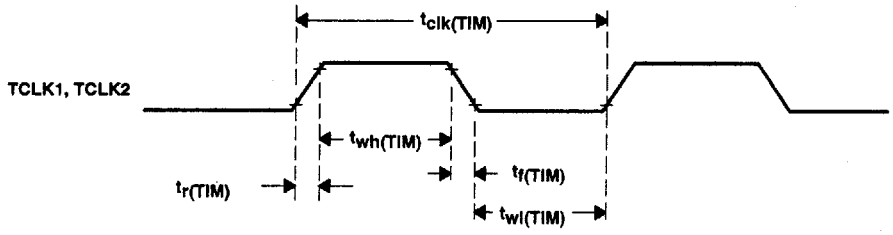
interrupt timing



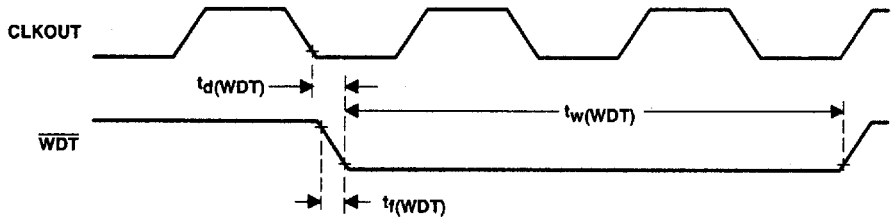
bit I/O timing



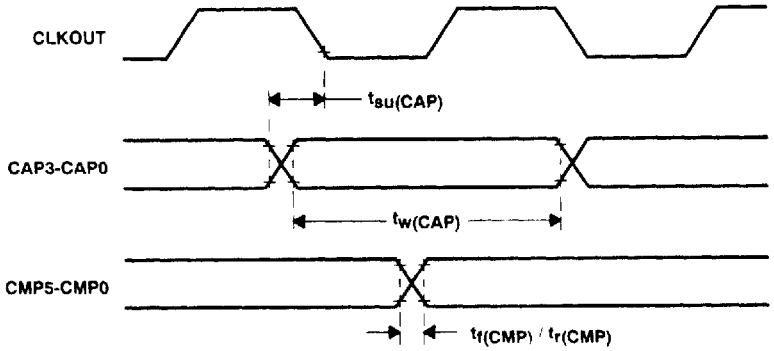
general purpose timers



watchdog timer



event manager



PROGRAMMING THE TMS320E14/P14 EPROM CELL

The 'E14 and 'P14 include a 4K × 16-bit industry-standard EPROM cell for prototyping and low-volume production. The 'C14 with a 4K-word masked ROM then provides a migration path for cost-effective production. An EPROM adapter socket (part # TMDX3270110), shown in Figure 5, is available to provide 68-pin to 28-pin conversion for programming the 'E14 and 'P14.

Key features of the EPROM cell include the normal programming operation as well as verification. The EPROM cell also includes a code protection feature that allows code to be protected against copyright violations.

The 'E14/P14 EPROM cells are programmed using the same family and device codes as the TMS27C64 8K × 8-bit EPROM. The TMS27C64 EPROM series are ultraviolet-light erasable, electrically programmable, read-only memories, fabricated using HVCMOS technology. They are pin compatible with existing 28-pin ROMs and EPROMs. These EPROMs operate from a 5-V supply in the read mode; however, a 12.5-V supply is needed for programming. All programming signals are TTL level. For programming outside the system, existing EPROM programmers can be used. Locations may be programmed singly, in blocks, or at random.

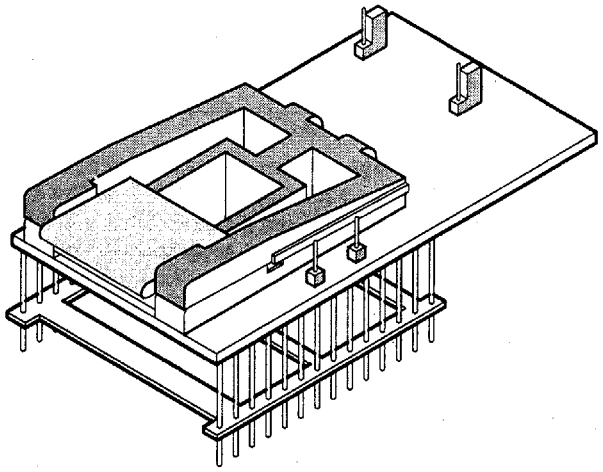


Figure 5. EPROM Adapter Socket

The 'E14/P14 devices use 13 address lines to address the 4K-word memory in byte format (8K-byte memory). In word format, the most-significant byte of each word is assigned an even address and the least-significant byte an odd address in the byte format. Programming information should be downloaded to EPROM programmer memory in a high-byte to low-byte order for proper programming of the devices (see Figure 6).

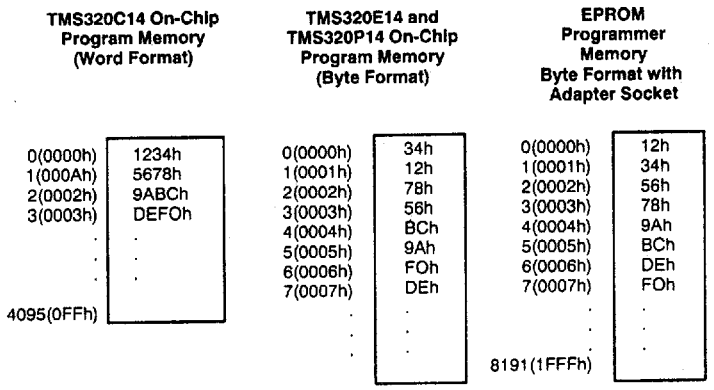


Figure 6. Programming Data Format

Figure 7 shows the wiring conversion to program the 'E14 and 'P14 using the 28-pin pinout of the TMS27C64. The table of pin nomenclature provides a description of the TMS27C64 pins.

CAUTION

The 'E14 and 'P14 do not support the signature mode available with some EPROM programmers. The signature mode places high voltage (12.5 V_{dc}) on pin A9. The 'E14 and 'P14 EPROM cells are not designed for this feature and will be damaged if subjected to it. A 3.9 kΩ resistor is standard on the TI programmer socket between pin A9 and programmer. This protects the device from unintentional use of the signature mode.

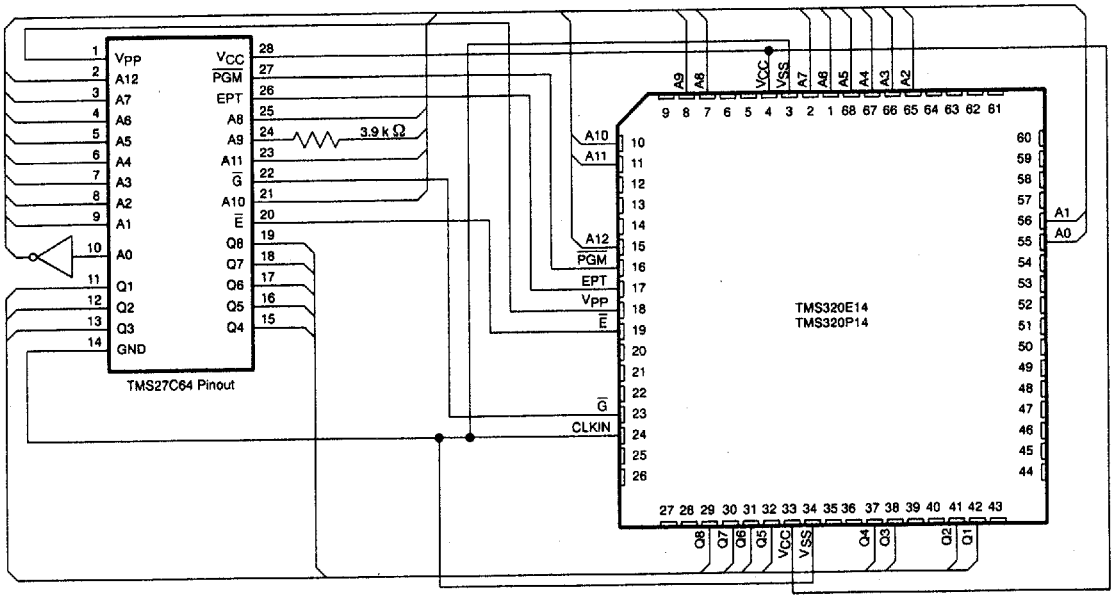


Figure 7. TMS320E14/P14 EPROM Programming Conversion to TMS27C64 EPROM Pinout

TERMINAL FUNCTIONS (TMS320E14/P14)

NAME	I/O	DEFINITION
A12(MSB)-A0(LSB)	I	On-chip EPROM programming address lines
CLKIN	I	Clock oscillator input
$\overline{E}$	I	EPROM chip enable
EPT	I	EPROM test mode select
$\overline{G}$	I	EPROM output enable
GND	I	Ground
$\overline{PGM}$	I	EPROM write/program select
Q8(MSB)-Q1(LSB)	I/O	Data lines for byte-wide programming of on-chip 8K bytes of EPROM
$\overline{RS}$	I	Reset for initializing the device
V_{CC}	I	5-V to 6.5-V power supply
V_{PP}	I	12.5-V to 13-V power supply

Table 4 shows the programming levels required for programming, verifying, reading, and protecting the EPROM cell.

Table 4. TMS320E14/P14 Programming Mode Levels

SIGNAL NAME†	TMS320E14/P14 PIN	TMS27C64 PIN	PROGRAM	PROGRAM VERIFY	READ	EPROM PROTECT	PROTECT VERIFY
$\overline{E}$	19	20	V_{IL}	V_{IL}	V_{IL}	V_{IH}	V_{IL}
$\overline{G}$	23	22	V_{IH}	$\overline{PULSE}$	$\overline{PULSE}$	V_{IH}	V_{IL}
$\overline{PGM}$	16	27	$\overline{PULSE}$	V_{IH}	V_{IH}	V_{IH}	V_{IH}
V_{PP}	18	1	V_{PP}	V_{PP}	V_{CC}	V_{PP}	V_{CCP}
V_{CC}	4,33	28	V_{CCP}	V_{CCP}	V_{CC}	V_{CCP}	V_{CCP}
V_{SS}	3,34	14	V_{SS}	V_{SS}	V_{SS}	V_{SS}	V_{SS}
CLKIN	24	14	V_{SS}	V_{SS}	V_{SS}	V_{SS}	V_{SS}
EPT	17	26	V_{SS}	V_{SS}	V_{SS}	V_{PP}	V_{PP}
Q1-Q8	42, 41, 38, 37, 32-29	11–13, 15-19,	Data In	Data Out	Data Out	$Q_8 = \overline{PULSE}$	Q_8 = RBIT
A12-A7	15, 11, 10, 8, 7, 2	2, 23, 21, 24, 25, 3	ADDR	ADDR	ADDR	X	X
A6	1	4	ADDR	ADDR	ADDR	X	V_{IL}
A5	68	5	ADDR	ADDR	ADDR	X	X
A4	67	6	ADDR	ADDR	ADDR	V_{IH}	X
A3-A0	66, 65, 56, 55	7-10	ADDR	ADDR	ADDR	X	X

† Signal names shown for 'E14/P14 EPROM programming mode only.

Legend:

V_{IH} = TTL high level; V_{IL} = TTL low level; ADDR = byte address bit; V_{PP} = 12.5 V ± 0.25 V (FAST) or 13 V ± 0.25 V (SNAP! Pulse).
V_{CC} = 5 V ± 0.25 V; X = don't care; $\overline{PULSE}$ = low-going TTL pulse.
D_{IN} = byte to be programmed at ADDR; Q_{OUT} = byte stored at ADDR.; RBIT = ROM protect bit
V_{CCP} = 6 V ± 0.25 V (FAST) or 6.5 V ± 0.25 V (SNAP! Pulse).

programming

Since every memory in the cell is at a logic high, the programming operation reprograms selected bits to low. Once the '320E14 is programmed, these bits can only be erased using ultraviolet light. The correct byte is placed on the data bus with V_{PP} set to the 12.5-V level. The $\overline{PGM}$ pin is then pulsed low to program in the zeros.

erasure

Before programming, the 'E14 must be erased by exposing it to ultraviolet light. The recommended minimum exposure dose (UV-intensity × exposure-time) is 15 W•s/cm^2. A typical 12-mW•s/cm^2, filterless UV lamp will erase the device in 21 minutes. The lamp should be located about 2 5 cm above the chip during erasure. After exposure, all bits are in the high state

verify/read

To verify correct programming, the EPROM cell can be read using either the verify or read line definitions shown in Table 5, assuming the inhibit bit (RBIT) has not been programmed

program inhibit

Programming may be inhibited by maintaining a high level input on the $\overline{E}$ pin or $\overline{PGM}$ pin.

standard programming procedure

Before programming, the 'E14 must first be completely erased. The device can then be programmed with the correct code. It is advisable to program unused sections with zeros as a further security measure. After the programming is complete, the code programmed into the cell should be verified. If the cell passes verification, the next step is to program the ROM protect bit (RBIT). Once the RBIT programming is verified, an opaque label should be placed over the window to protect the EPROM cell from inadvertent erasure by ambient light. At this point, the programming is complete, and the device is ready to be placed into its destination circuit.

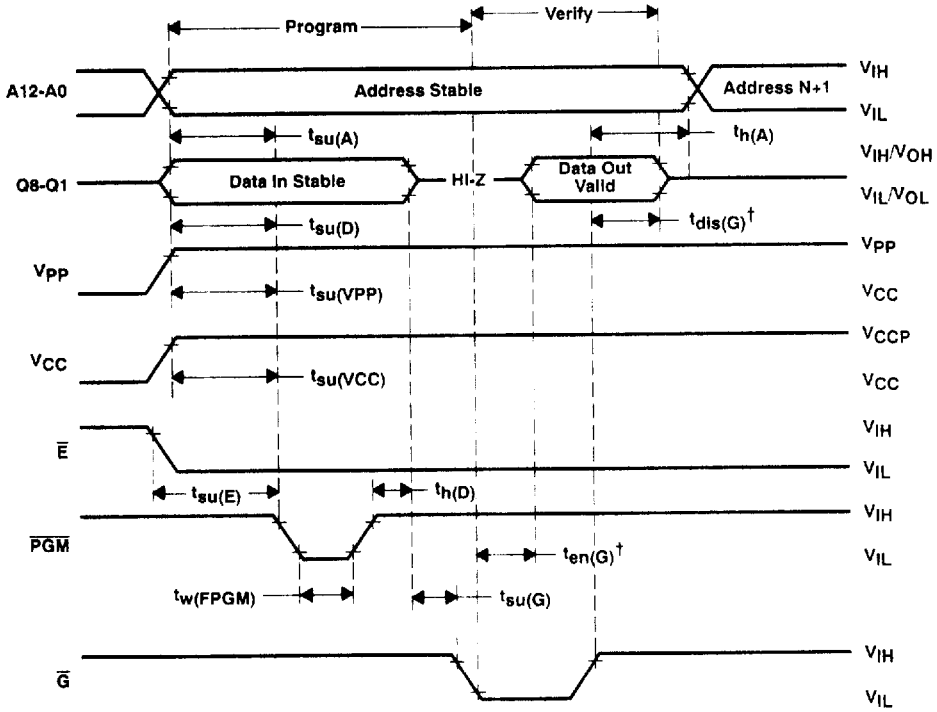
Refer to other appendices of the *TMS320C1x User's Guide* for additional information on EPROM programming.

recommended timing requirements for programming: V_{CC} = 6 V and V_{PP} = 12.5 V (FAST) or V_{CC} = 6.5 V and V_{PP} = 13 V (SNAP! PULSE), T_A = 25°C (see Note 13)

			MIN	NOM	MAX	UNIT
$t_{w(PGM)}$	*Initial program pulse duration*	Fast programming algorithm	0.95	1	1.05	ms
		SNAP! Pulse programming algorithm	95	100	105	µs
$t_{w(FPGM)}$	Final pulse duration	Fast programming only	2.85		78.75	ms
$t_{su(A)}$	Address setup time		2			µs
$t_{su(E)}$	$\overline{E}$ setup time		2			µs
$t_{su(G)}$	$\overline{G}$ setup time		2			µs
$t_{su(D)}$	Data setup time		2			µs
$t_{su(VPP)}$	V_{PP} setup time		2			µs
$t_{su(VCC)}$	V_{CC} setup time		2			µs
$t_{h(A)}$	Address hold time		0			µs
$t_{h(D)}$	Data hold time		2			µs

NOTE 13: For all switching characteristics and timing measurements, input pulse levels are 0 4 V to 2.4 V and V_{PP} = 12.5 V ± 0.5 V during programming.

program cycle timing

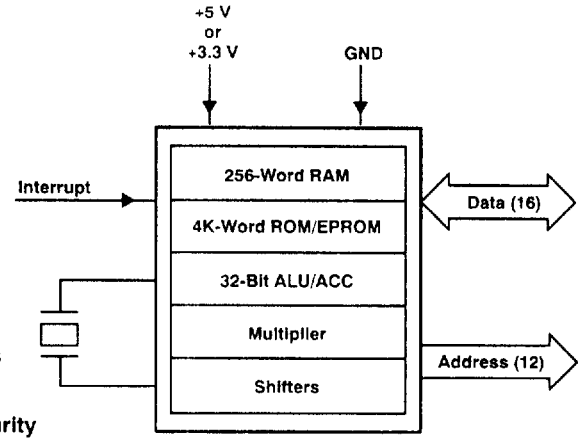


† $t_{dis(G)}$ and $t_{en(G)}$ are characteristics of the device but must be accommodated by the programmer.

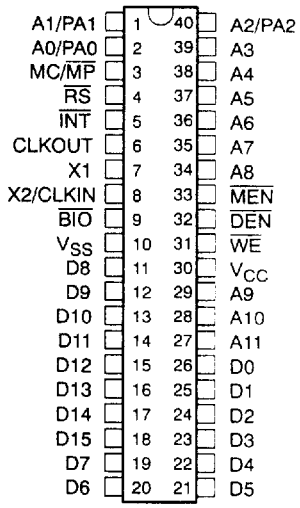
JANUARY 1987 — REVISED JULY 1991

Key Features: TM320C15/E15/LC15/P15

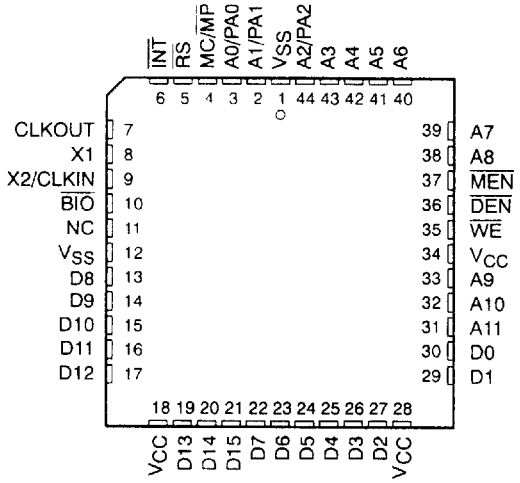
- Instruction Cycle Timing:
  - 160-ns (TMS320C15-25/E15-25)
  - 200-ns (TMS320C15/E15/P15)
  - 250-ns (TMS320LC15)
- 256 Words of On-Chip Data RAM
- 4K Words of On-Chip Program ROM (TMS320C15/C15-25/LC15)
- 4K Words of On-Chip Program EPROM (TMS320E15/E15-25)
- One-Time Programmable (OTP) Windowless EPROM Version Available (TMS320P15)
- EPROM Code Protection for Copyright Security
- External Memory up to 4K-Words at Full Speed
- 16 × 16-Bit Multiplier With 32-Bit Product
- 0 to 16-Bit Barrel Shifter
- On-Chip Clock Oscillator
- 3.3-V Low-Power Version Available (TMS320LC15)
- Device Packaging:
  - 40-Pin Dip (All Devices)
  - 44-Lead PLCC (TMS320C15/C15-25/LC15/P15)
  - 44-Lead-QUAD (TMS320E15/E15-25)



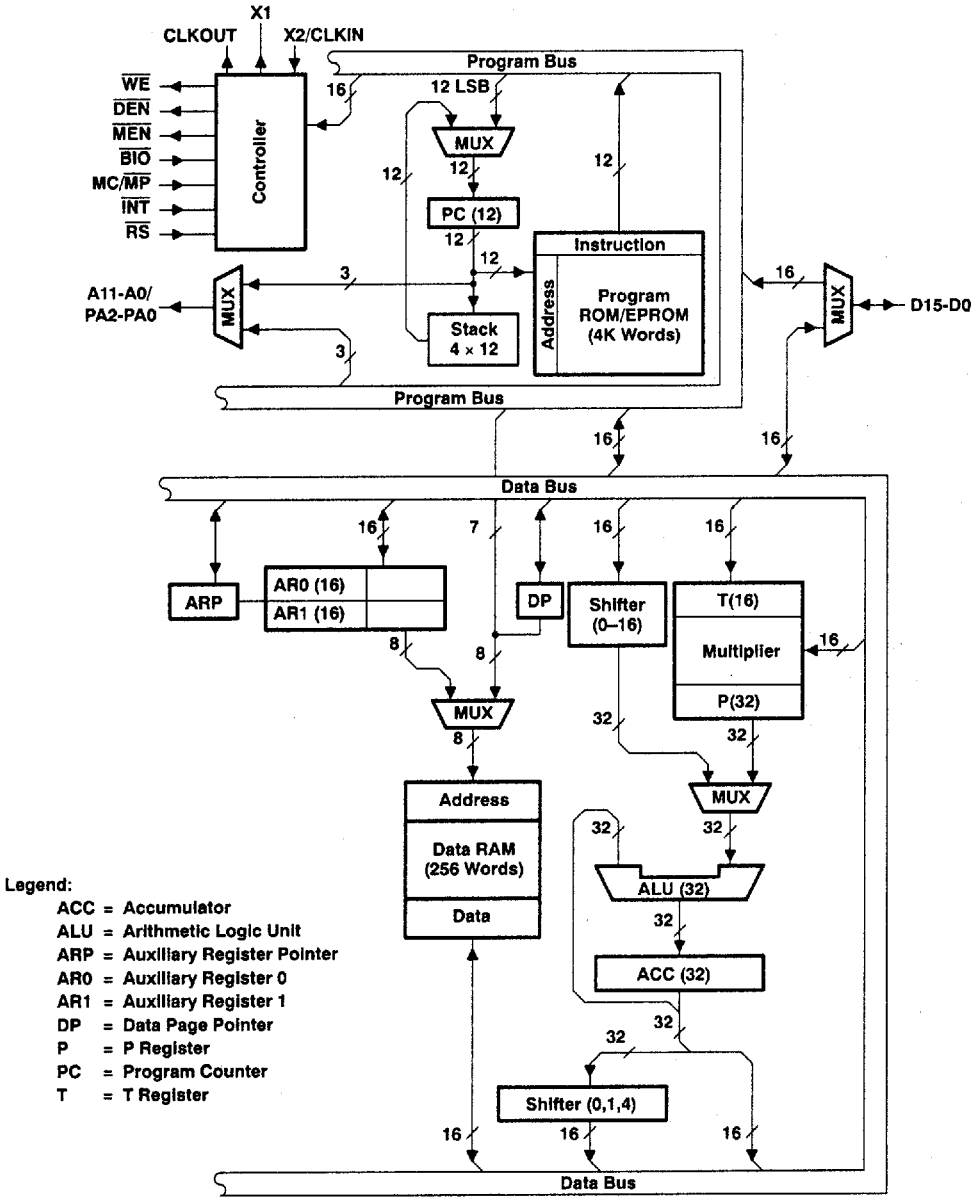
TMS320C15/E15/LC15/P15
N/JD Package
(Top View)



TMS320C15/E15/LC15/P15
FN/FZ Package
(Top View)



functional block diagram



Legend:

ACC = Accumulator
ALU = Arithmetic Logic Unit
ARP = Auxiliary Register Pointer
AR0 = Auxiliary Register 0
AR1 = Auxiliary Register 1
DP = Data Page Pointer
P = P Register
PC = Program Counter
T = T Register

TERMINAL FUNCTIONS (TMS320C15/E15/LC15/P15)†

NAME	I/O‡	DEFINITION
A11-A0/PA2-PA0	O	External address bus. I/O port address multiplexed over PA2-PA0.
$\overline{BIO}$	I	External polling input
CLKOUT	O	System clock output, 1/4 crystal/CLKIN frequency
D15-D0	I/O	16-bit parallel data bus
$\overline{DEN}$	O	Data enable for device input data on D15-D0
$\overline{INT}$	I	External interrupt input
MC/$\overline{MP}$	I	Memory mode select pin. High selects microcomputer mode. Low selects microprocessor mode.
$\overline{MEN}$	O	Memory enable indicates that D15-D0 will accept external memory instruction.
NC	O	No connection
$\overline{RS}$	I	Reset for initializing the device
V_{CC}	I	+ 5 V supply
V_{SS}	I	Ground
$\overline{WE}$	O	Write enable for device output data on D15-D0
X1	O	Crystal output for internal oscillator
X2/CLKIN	I	Crystal input internal oscillator or external system clock input

† See EPROM programming section.
‡ Input/Output/High-impedance state.

electrical specifications

This section contains the electrical specifications for the 'C15/E15/P15 digital signal processors, including test parameter measurement information. Parameters with $_{PP}$ subscripts apply only to the 'E15/P15 in the EPROM programming mode (see Note 11).

absolute maximum ratings over operating free-air temperature range (unless otherwise noted)†

Supply voltage range, V_{CC} (see Note 6) ..... – 0.3 V to 7 V
Supply voltage range, V_{PP} ..... – 0.6 V to 14 V
Input voltage range ..... – 0.3 V to 14 V
Output voltage range ..... – 0.3 V to 7 V
Continuous power dissipation ..... 0.5 mW
Operating free-air temperature: L suffix ..... 0°C to 70°C
A suffix ..... – 40°C to 85°C
Storage temperature ..... – 55 °C to 150 °C

† Stresses beyond those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. This is a stress rating only, and functional operation of the device at these or any other conditions beyond those indicated in the "Recommended Operating Conditions" section of this specification is not implied. Exposure to absolute-maximum-rated conditions for extended periods may affect device reliability.

NOTE 6: All voltage values are with respect to V_{SS}.

recommended operating conditions

			MIN	NOM	MAX	UNIT
V_{CC}	Supply voltage	EPROM devices	4.75	5	5.25	V
		All other devices	4.5	5	5.5	V
V_{PP}	Supply voltage (see Note 11)		12.25	12.5	12.75	V
V_{SS}	Supply voltage			0		V
V_{IH}	High-level input voltage	CLKIN	3			V
		All remaining inputs	2			V
V_{IL}	Low-level input voltage	MC/$\overline{MP}$			0.6	V
		All remaining inputs			0.8	V
I_{OH}	High-level output current, all outputs				– 300	μA
I_{OL}	Low-level output current (All outputs except for TMS320LC15)				2	mA
T_A	Operating free-air temperature	L suffix	0		70	°C
		A suffix	– 40		85	°C

NOTE 11: V_{PP} can be applied only to programming pins designed to accept V_{PP} as an input. During programming the total supply current is $I_{PP} + I_{CC}$.

electrical characteristics over specified temperature range (unless otherwise noted)

PARAMETER		TEST CONDITIONS		MIN	TYP†	MAX	UNIT
V_{OH} High-level output voltage		I_{OH} = MAX		2.4	3		V
		I_{OH} = 20 μA (see Note 8)		V_{CC} – 0.4			V
V_{OL} Low-level output voltage		I_{OL} = MAX			0.3	0.5	V
I_{OZ} Off-state output current		V_{CC} = MAX	V_O = 2.4 V			20	μA
			V_O = 0.4 V			– 20	
I_I Input current		V_I = V_{SS} to V_{CC}	All inputs except CLKIN			±20	μA
			CLKIN			±50	
I_{CC}‡ Supply current	TMS320C15	f = 20.5 MHz, V_{CC} = 5.5 V, T_A = 0°C to 70°C			45	55	mA
	TMS320C15-25	f = 25.6 MHz, V_{CC} = 5.5 V, T_A = 0°C to 70°C			50	65	
	TMS320E15	f = 20.5 MHz, V_{CC} = 5.25 V, T_A = – 40°C to 85°C			55	75	
	TMS320E15-25	f = 25.6 MHz, V_{CC} = 5.25 V, T_A = 0°C to 70°C			65	85	
C_i Input capacitance	Data bus	f = 1 MHz, all other pins 0 V			25‡		pF
	All other				15‡		
C_o Output capacitance	Data bus				25‡		pF
	All others				10‡		

† All typical values are at V_{CC} = 5 V, T_A = 70°C and are used for thermal resistance calculations.
‡ I_{CC} characteristics are inversely proportional to temperature. For I_{CC} dependence on temperature, frequency, and loading, see Figure 3.
NOTE 7: This voltage specification is included for interface to HC logic. However, note that all of the other timing parameters defined in this data sheet are specified for TTL logic levels and will differ for HC logic levels.

CLOCK CHARACTERISTICS AND TIMING

The TMS320C15/E15/P15 can use either its internal oscillator or an external frequency source for a clock.

internal clock option

The internal oscillator is enabled by connecting a crystal across X1 and X2/CLKIN (see Figure 1). The frequency of CLKOUT is one-fourth the crystal fundamental frequency. The crystal should be fundamental mode, and parallel resonant, with an effective series resistance of 30 ohms, a power dissipation of 1 mW, and should be specified at a load capacitance of 20 pF.

PARAMETER		TEST CONDITIONS	MIN	NOM	MAX	UNIT
Crystal frequency, f_x	TMS320C15	T_A = 0°C to 70°C	6.7		20.5	MHz
	TMS320E15/P15	T_A = – 40°C to 85°C	6.7		20.5	
	TMS320C15-25/E15-25	T_A = 0°C to 70°C	6.7		25.6	
C1, C2		T_A = 0°C to 70°C		10		pF

external clock option

An external frequency source can be used by injecting the frequency directly into X2/CLKIN with X1 left unconnected. The external frequency injected must conform to the specifications listed in the table below.

switching characteristics over recommended operating conditions

PARAMETER		TEST CONDITIONS	TMS320C15/E15/P15			TMS320C15-25/E15-25			UNIT
			MIN	NOM	MAX	MIN	NOM	MAX	
$t_{c(C)}$	CLKOUT cycle time‡	R_L = 825 Ω, C_L = 100 pF (see Figure 2)	195.12	200		156.25	160		ns
$t_{r(C)}$	CLKOUT rise time			10†			10†		ns
$t_{f(C)}$	CLKOUT fall time			8†			8†		ns
$t_{w(CL)}$	Pulse duration, CLKOUT low			92†			72†		ns
$t_{w(CH)}$	Pulse duration, CLKOUT high			90†			70†		ns
$t_{d(MCC)}$	Delay time, CLKIN↑ to CLKOUT↓		25†		60†	25†		50†	ns

† Values derived from characterization data and not tested.
‡ $t_{c(C)}$ is the cycle time of CLKOUT, i.e., $4t_{c(MC)}$ (4 times CLKIN cycle time if an external oscillator is used).

timing requirements over recommended operating conditions

		TMS320C15/E15/P15			TMS320C15-25/E15-25			UNIT
		MIN	NOM	MAX	MIN	NOM	MAX	
$t_{c(MC)}$	Master clock cycle time	48.78	50	150	39.06	40	150	ns
$t_{r(MC)}$	Rise time, master clock input		5†	10†		5†	10†	ns
$t_{f(MC)}$	Fall time, master clock input		5†	10†		5†	10†	ns
$t_{w(MCP)}$†	Pulse duration, master clock	$0.4t_{c(MC)}$		$0.6t_{c(MC)}$†	$0.45t_{c(MC)}$		$0.55t_{c(MC)}$†	ns
$t_{w(MCL)}$	Pulse duration, master clock low		20†			15†		ns
$t_{w(MCH)}$	Pulse duration, master clock high		20†			15†		ns

† Values derived from characterization data and not tested.

MEMORY AND PERIPHERAL INTERFACE TIMING

switching characteristics over recommended operating conditions

PARAMETER		TEST CONDITIONS	TMS320C15/E15/P15			TMS320C15-25/E15-25			UNIT
			MIN	NOM	MAX	MIN	NOM	MAX	
t_{d1}	Delay time, CLKOUT↓ to address bus valid	R_L = 825 Ω, C_L = 100 pF (see Figure 2)	10†		50	10‡		40	ns
t_{d2}	Delay time, CLKOUT↓ to $\overline{MEN}$↓		$1/4t_{c(C)} - 5$†		$1/4t_{c(C)} + 15$	$1/4t_{c(C)} - 5$†		$1/4t_{c(C)} + 12$	ns
t_{d3}	Delay time, CLKOUT↓ to $\overline{MEN}$↑		−10†		15	−10†		12	ns
t_{d4}	Delay time, CLKOUT↓ to $\overline{DEN}$↓		$1/4t_{c(C)} - 5$†		$1/4t_{c(C)} + 15$	$1/4t_{c(C)} - 5$†		$1/4t_{c(C)} + 12$	ns
t_{d5}	Delay time, CLKOUT↓ to $\overline{DEN}$↑		−10†		15	−10†		12	ns
t_{d6}	Delay time, CLKOUT↓ to $\overline{WE}$↓		$1/2t_{c(C)} - 5$†		$1/2t_{c(C)} + 15$	$1/2t_{c(C)} - 5$†		$1/2t_{c(C)} + 12$	ns
t_{d7}	Delay time, CLKOUT↓ to $\overline{WE}$↑		−10†		15	−10†		12	ns
t_{d8}	Delay time, CLKOUT↓ to data bus OUT valid				$1/4t_{c(C)} + 65$			$1/4t_{c(C)} + 52$	ns
t_{d9}	Time after CLKOUT↓ that data bus starts to be driven		$1/4t_{c(C)} - 5$†			$1/4t_{c(C)} - 5$†			ns
t_{d10}	Time after CLKOUT↓ that data bus stops being driven (TMS320C15/C15-25 only)				$1/4t_{c(C)} + 40$†			$1/4t_{c(C)} + 40$†	ns
t_{d10}	Time after CLKOUT↓ that data bus stops being driven (TMS320E15/E15-25 only)				$1/4t_{c(C)} + 70$†			$1/4t_{c(C)} + 70$†	ns
t_v	Data bus OUT valid after CLKOUT↓		$1/4t_{c(C)} - 10$			$1/4t_{c(C)} - 10$			ns
$t_{h(A\text{-}WMD)}$	Address hold time after $\overline{WE}$↑, $\overline{MEN}$↑, or $\overline{DEN}$↑ (see Note 15)		0†		2†	0†		2†	ns
$t_{su(A\text{-}MD)}$	Address bus setup time prior to $\overline{DEN}$↓		$1/4t_{c(C)} - 45$			$1/4t_{c(C)} - 35$			ns

† Values derived from characterization data and not tested.
NOTE 14: Address bus will be valid upon $\overline{WE}$↑, $\overline{MEN}$↑, or $\overline{DEN}$↑.

timing requirements over recommended operating conditions

		TEST CONDITIONS	TMS320C15/E15/P15			TMS320C15-25/E15-25			UNIT
			MIN	NOM	MAX	MIN	NOM	MAX	
$t_{su(D)}$	Setup time, data bus valid prior to CLKOUT↓	R_L = 825 Ω, C_L = 100 pF (see Figure 2)	50			40			ns
$t_{h(D)}$	Hold time, data bus held valid after CLKOUT↓ (see Note 9)		0			0			ns

NOTE 9: Data may be removed from the data bus upon $\overline{MEN}$↑ or $\overline{DEN}$↑ preceding CLKOUT↓.

RESET ($\overline{RS}$) TIMING

switching characteristics over recommended operating conditions

PARAMETER		TEST CONDITIONS	MIN	TYP	MAX	UNIT
t_{d11}	Delay time, $\overline{DEN}\uparrow$, $\overline{WE}\uparrow$, and $\overline{MEN}\uparrow$ from $\overline{RS}$	R_L = 825 Ω, C_L = 100 pF (see Figure 2)			$1/2t_{c(C)}+50^\dagger$	ns
$t_{dis(R)}$	Data bus disable time after $\overline{RS}$				$1/4t_{c(C)}+50^\dagger$	ns

† Values derived from characterization data and not tested.

timing requirements over recommended operating conditions

		TMS320C15/E15/P15			TMS320C15-25/E15-25			UNIT
		MIN	NOM	MAX	MIN	NOM	MAX	
$t_{su(R)}$	Reset ($\overline{RS}$) setup time prior to CLKOUT (see Note 10)	50			40			ns
$t_{w(R)}$	$\overline{RS}$ pulse duration	$5t_{c(C)}$			$5t_{c(C)}$			ns

NOTE 10: $\overline{RS}$ can occur anytime during a clock cycle. Time given is minimum to ensure synchronous operation.

INTERRUPT ($\overline{INT}$) TIMING

timing requirements over recommended operating conditions

		TMS320C15/E15/P15			TMS320C15-25/E15-25			UNIT
		MIN	NOM	MAX	MIN	NOM	MAX	
$t_{f(INT)}$	Fall time, $\overline{INT}$			15			15	ns
$t_{w(INT)}$	Pulse duration, $\overline{INT}$	$t_{c(C)}$			$t_{c(C)}$			ns
$t_{su(INT)}$	Setup time, $\overline{INT}\downarrow$ before CLKOUT↓	50			40			ns

IO ($\overline{BIO}$) TIMING

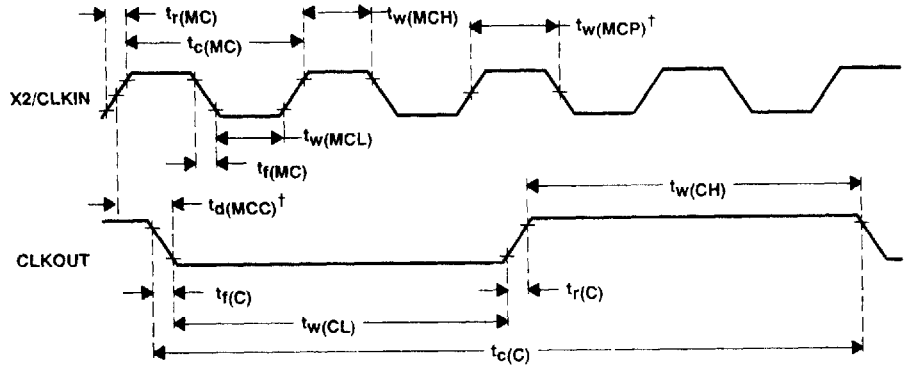
timing requirements over recommended operating conditions

		TMS320C15/E15/P15			TMS320C15-25/E15-25			UNIT
		MIN	NOM	MAX	MIN	NOM	MAX	
$t_{f(IO)}$	Fall time, $\overline{BIO}$			15			15	ns
$t_{w(IO)}$	Pulse duration, $\overline{BIO}$	$t_{c(C)}$			$t_{c(C)}$			ns
$t_{su(IO)}$	Setup time, $\overline{BIO}\downarrow$ before CLKOUT↓	50			40			ns

TIMING DIAGRAMS

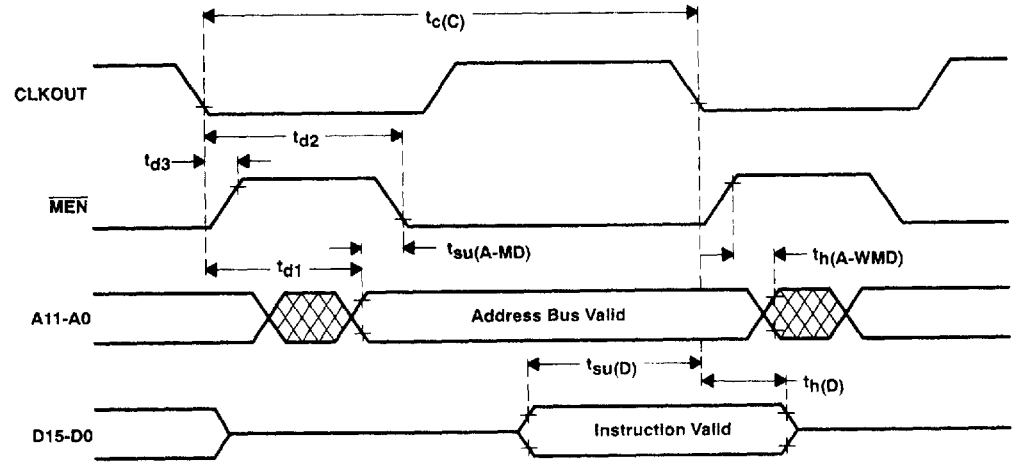
Timing measurements are referenced to and from a low voltage of 0.8 volts and a high voltage of 2.0 volts, unless otherwise noted.

clock timing

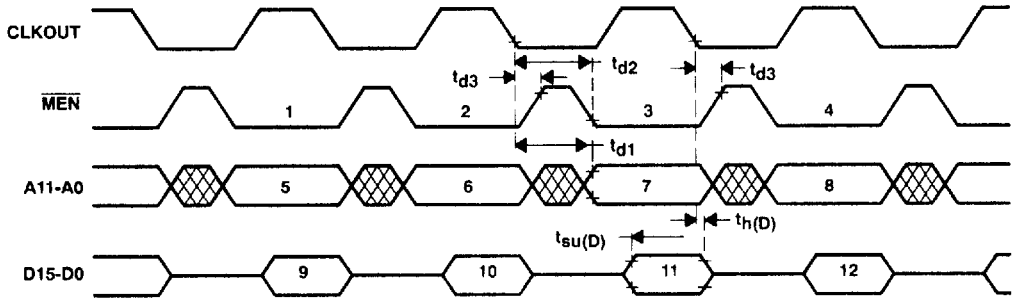


† $t_{d(MCC)}$ and $t_{w(MCP)}$ are referenced to an intermediate level of 1.5 V on the CLKIN waveform.

memory read timing



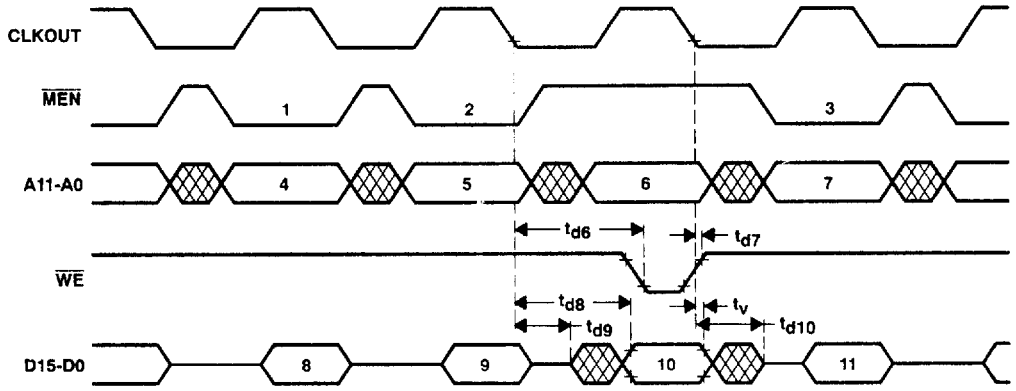
TBLR instruction timing



Legend:

1. TBLR Instruction Prefetch
2. Dummy Prefetch
3. Data Fetch
4. Next Instruction Prefetch
5. Address Bus Valid
6. Address Bus Valid
7. Address Bus Valid
8. Address Bus Valid
9. Instruction Valid
10. Instruction Valid
11. Data Input Valid
12. Instruction Valid

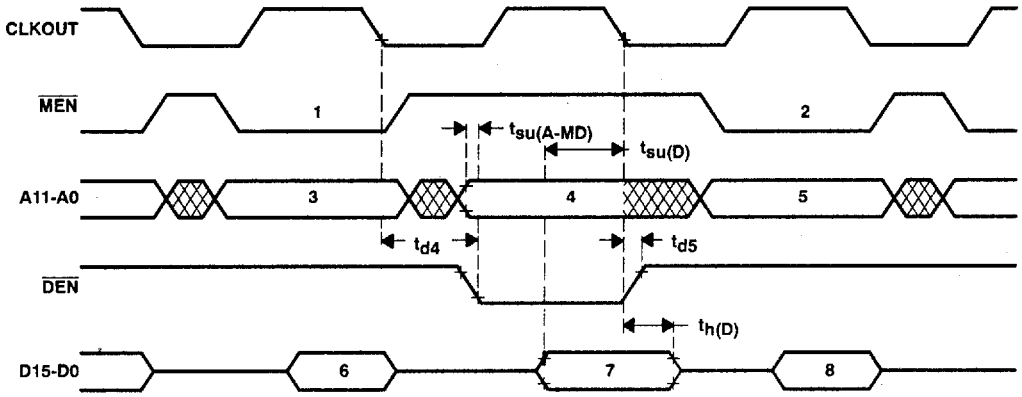
TBLW instruction timing



Legend:

1. TBLW Instruction Prefetch
2. Dummy Prefetch
3. Next Instruction Prefetch
4. Address Bus Valid
5. Address Bus Valid
6. Address Bus Valid
7. Address Bus Valid
8. Instruction Valid
9. Instruction Valid
10. Data Output Valid
11. Instruction Valid

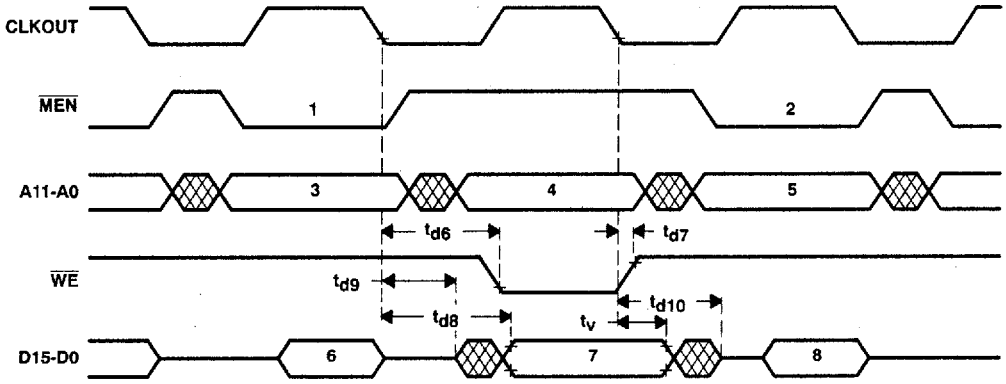
IN instruction timing



Legend:

1. IN Instruction Prefetch
2. Next Instruction Prefetch
3. Address Bus Valid
4. Peripheral Address Valid
5. Address Bus Valid
6. Instruction Valid
7. Data Input Valid
8. Instruction Valid

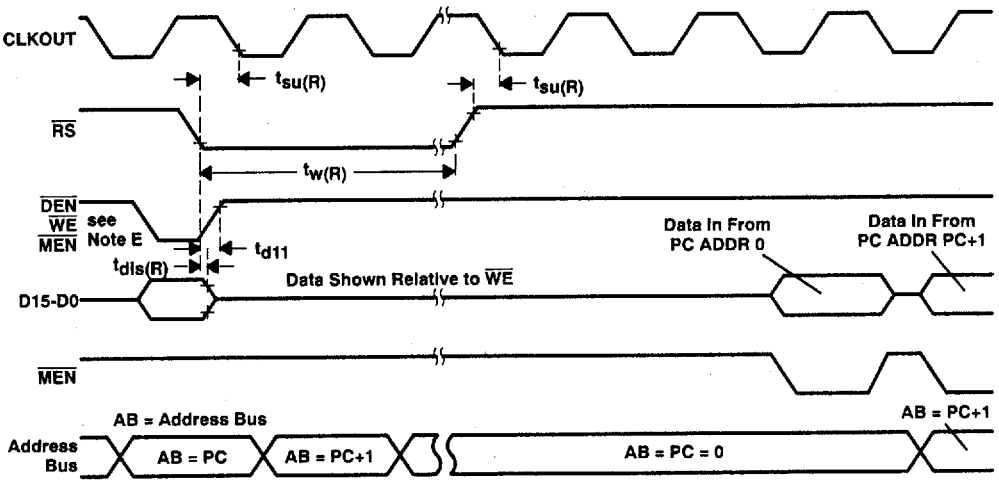
OUT instruction timing



Legend:

1. IN Instruction Prefetch
2. Next Instruction Prefetch
3. Address Bus Valid
4. Peripheral Address Valid
5. Address Bus Valid
6. Instruction Valid
7. Data Output Valid
8. Instruction Input Valid

reset timing



NOTES: A. $\overline{RS}$ forces $\overline{DEN}$, $\overline{WE}$, and $\overline{MEN}$ high and places data bus D0 through D15 in a high-impedance state. AB outputs (and program counter) are synchronously cleared to zero after the next complete CLK cycle from $\overline{RS}\downarrow$.

B. $\overline{RS}$ must be maintained for a minimum of five clock cycles.

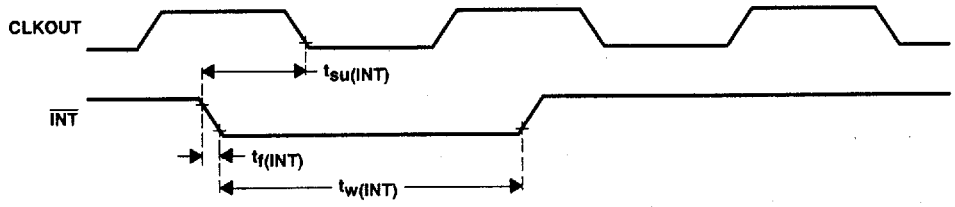
C. Resumption of normal program will commence after one complete CLK cycle from $\overline{RS}\uparrow$.

D. Due to the synchronization action on $\overline{RS}$, time to execute the function can vary dependent upon when $\overline{RS}\uparrow$ or $\overline{RS}\downarrow$ occur in the CLK cycle.

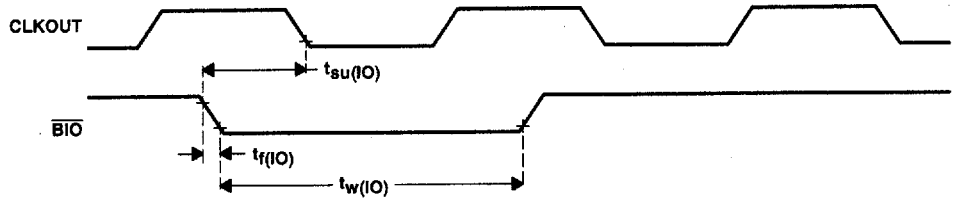
E. Diagram shown is for definition purpose only. $\overline{DEN}$, $\overline{WE}$, and $\overline{MEN}$ are mutually exclusive.

F. During a write cycle, $\overline{RS}$ may produce an invalid write address.

Interrupt timing



$\overline{BIO}$ timing



absolute maximum ratings over specified temperature range (unless otherwise noted)†

Supply voltage range, V_{PP} (see Note 6) ..... – 0.6V to 14 V

† Stresses beyond those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. This is a stress rating only, and functional operation of the device at these or any other conditions beyond those indicated in the "Recommended Operating Conditions" section of this specification is not implied. Exposure to absolute-maximum-rated conditions for extended periods may affect device reliability.

NOTE 6: All voltage values are with respect to V_{SS}.

recommended operating conditions

		MIN	NOM	MAX	UNIT
V_{PP}	Supply voltage (see Note 11)	12.25	12.5	12.75	V

NOTE 11: V_{PP} can be applied only to programming pins designed to accept V_{PP} as an input. During programming the total supply current is $I_{PP} + I_{CC}$.

electrical characteristics over specified temperature range (unless otherwise noted)

PARAMETER		TEST CONDITIONS	MIN	TYP‡	MAX	UNIT
I_{PP1}	V_{PP} supply current	$V_{PP} = V_{CC}$ 5.5 V			100	V
I_{PP2}	V_{PP} supply current (during program pulse)	V_{PP} = 12.75 V		30	50	V

‡ All typical values except for I_{CC} are at V_{CC} = 5 V, T_A = 25°C.

recommended timing requirements for programming, T_A = 25°C, V_{CC} = 6, V_{PP} = 12.5 V, (see Note 13)

		MIN	NOM	MAX	UNIT
$t_{w(IPGM)}$	Initial program pulse duration	0.95	1	1.05	ms
$t_{w(FPGM)}$	Final pulse duration	3.8		63	ms
$t_{su(A)}$	Address setup time	2			µs
$t_{su(E)}$	$\overline{E}$ setup time	2			µs
$t_{su(G)}$	$\overline{G}$ setup time	2			µs
$t_{dis(G)}$	Output disable time from $\overline{G}$ (see Note 15)	0		130§	ns
$t_{en(G)}$	Output enable time from $\overline{G}$	0		150§	ns
$t_{su(D)}$	Data setup time	2			µs
$t_{su(VPP)}$	V_{PP} setup time	2			µs
$t_{su(VCC)}$	V_{CC} setup time	2			µs
$t_{h(A)}$	Address hold time	0			µs
$t_{h(D)}$	Data hold time	2			µs

§ Values derived from characterization data and not tested.

NOTES: 13. For all switching characteristics and timing measurements, input pulse levels are 0.4 V to 2.4 V and V_{PP} = 12.5 V ± 0.5 V during programming.

15. Common test conditions apply for $t_{dis(G)}$ except during programming.

PROGRAMMING THE TMS320E15/P15 EPROM CELL

'E15/P15 devices include a 4K x 16-bit industry-standard EPROM cell for prototyping, early field testing, and low-volume production. In conjunction with this EPROM, the 'E15/P15 with a 4K-word masked ROM, then, provide more migration paths for cost-effective production.

EPROM adapter sockets are available that provide pin-to-pin conversions for programming any 'E15/P15 devices. One adapter socket (part number RTC/PGM320C-06), shown in Figure 8, converts a 40-pin DIP device into an equivalent 28-pin device. Another socket (part number RTC/PGM320A-06), not shown, permits 44- to 28-pin conversion.

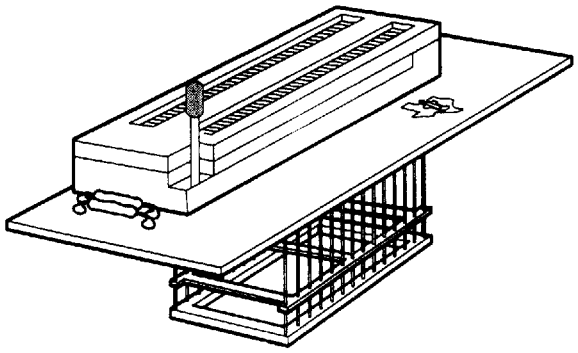
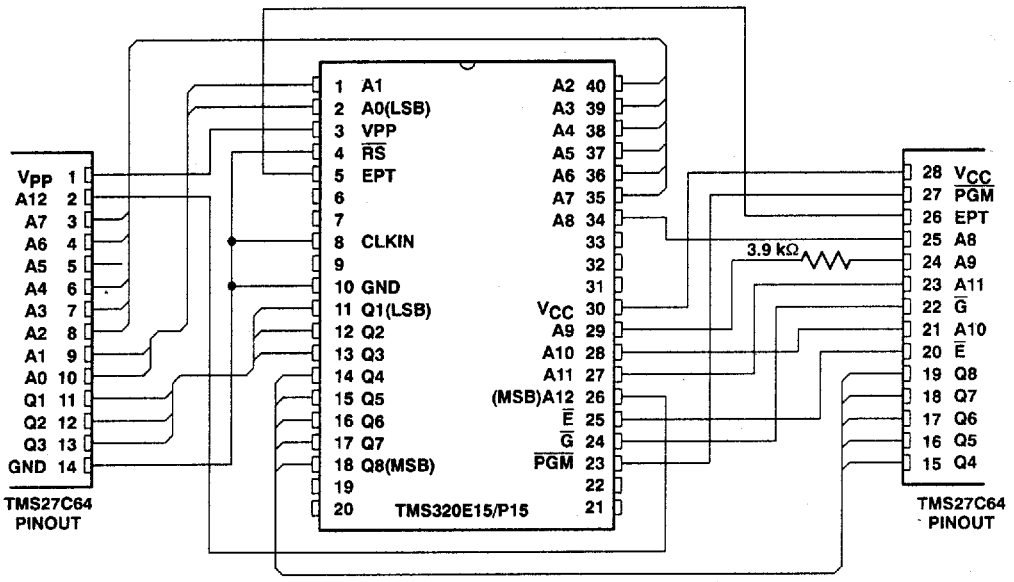


Figure 8. EPROM Adapter Socket (40-pin to 28-pin DIP Conversion)

Key features of the EPROM cell include the normal programming operation as well as verification. The EPROM cell also includes a code protection feature that allows code to be protected against copyright violations.

The 'E15/P15 EPROM cell is programmed using the same family and device pinout codes as the TMS27C64 8K x 8-bit EPROM. The TMS27C64 EPROM series are ultraviolet-light erasable, electrically programmable, read-only memories, fabricated using HVCMOS technology. They are pin-compatible with existing 28-pin ROMs and EPROMs. These EPROMs operate from a single 5-V supply in the read mode; however, a 12.5-V supply is needed for programming. All programming signals are TTL level. For programming outside the system, existing EPROM programmers can be used. Locations may be programmed singly, in blocks, or at random.

Figure 9 shows the wiring conversion to program the 'E15/P15 using the 28-pin pinout of the TMS27C64. Table 5 on pin nomenclature provides a description of the TMS27C64 pins. The code to be programmed into the device should be in serial mode. The 'E15/P15 devices use 13 address lines to address 4K-word memory in byte format.



CAUTION

Although acceptable by some EPROM programmers, the signature mode cannot be used on any 'E1x device. The signature mode will input a high-level voltage (12.5 V_{dc}) onto pin A9. Since this pin is not designed for high voltage, the cell will be damaged. To prevent an accidental application of voltage, Texas Instruments has inserted a 3.9 kΩ resistor between pin A9 of the TI programmer socket and the programmer itself.

Pin Nomenclature (TMS320E15/P15)

NAME	I/O	DEFINITION
A0-A12	I	On-chip EPROM programming address lines
CLKIN	I	Clock oscillator input
$\overline{E}$	I	EPROM chip select
EPT	I	EPROM test mode select
$\overline{G}$	I	EPROM read/verify select
GND	I	Ground
$\overline{PGM}$	I	EPROM write/program select
Q1-Q8	I/O	Data lines for byte-wide programming of on-chip 8K bytes of EPROM
$\overline{RS}$	I	Reset for initializing the device
V_{CC}	I	5-V power supply
V_{PP}	I	12.5-V power supply

Figure 9. TMS320E15/P15 EPROM Programming Conversion to TMS27C64 EPROM Pinout

Table 5 shows the programming levels required for programming, verifying, reading, and protecting the EPROM cell.

Table 5. TMS320E15/P15 Programming Mode Levels

SIGNAL NAME	TMS320E15 PIN	TMS27C64 PIN	PROGRAM	VERIFY	READ	PROTECT VERIFY	EPROM PROTECT
$\overline{E}$	25	20	V_{IL}	V_{IL}	V_{IL}	V_{IL}	V_{IH}
$\overline{G}$	24	22	V_{IH}	$\overline{PULSE}$	$\overline{PULSE}$	V_{IL}	V_{IH}
$\overline{PGM}$	23	27	$\overline{PULSE}$	V_{IH}	V_{IH}	V_{IH}	V_{IH}
V_{PP}	3	1	V_{PP}	V_{PP}	V_{CC}	V_{CC} + 1	V_{PP}
V_{CC}	30	28	V_{CC}	V_{CC}	V_{CC}	V_{CC} + 1	V_{CC} + 1
V_{SS}	10	14	V_{SS}	V_{SS}	V_{SS}	V_{SS}	V_{SS}
CLKIN	8	14	V_{SS}	V_{SS}	V_{SS}	V_{SS}	V_{SS}
$\overline{RS}$	4	14	V_{SS}	V_{SS}	V_{SS}	V_{SS}	V_{SS}
EPT	5	26	V_{SS}	V_{SS}	V_{SS}	V_{PP}	V_{PP}
Q1-Q8	11-18	11-13, 15-19	D_{IN}	Q_{OUT}	Q_{OUT}	Q8=RBIT	Q8=$\overline{PULSE}$
A0-A3	2, 1, 40, 39	10-7	ADDR	ADDR	ADDR	X	X
A4	38	6	ADDR	ADDR	ADDR	X	V_{IH}
A5	37	5	ADDR	ADDR	ADDR	X	X
A6	36	4	ADDR	ADDR	ADDR	V_{IL}	X
A7-A9	35, 34, 29	3, 25, 24	ADDR	ADDR	ADDR	X	X
A10-A12	28-26	21, 23, 2	ADDR	ADDR	ADDR	X	X

Legend:

V_{IH} = TTL high level; V_{IL} = TTL low level; ADDR = byte address bit
V_{PP} = 12.5 V ± 0.25 V; V_{CC} = 5 V ± 0.25 V; X = don't care
$\overline{PULSE}$ = low-going TTL level pulse; D_{IN} = byte to be programmed at ADDR
Q_{OUT} = byte stored at ADDR; RBIT = ROM protect bit.

programming

Since every memory bit in the cell is a logic 1, the programming operation reprograms certain bits to 0. Once programmed, these bits can only be erased using ultraviolet light. The correct byte is placed on the data bus with V_{PP} set to the 12.5 V level. The $\overline{PGM}$ pin is then pulsed low to program in the zeros.

erasure

Before programming, the device must be erased by exposing it to ultraviolet light. The recommended minimum exposure dose (UV-intensity × exposure-time) is 15 W•s/cm^2. A typical 12-mW/cm^2, filterless UV lamp will erase the device in 21 minutes. The lamp should be located about 2.5 cm above the chip during erasure. After exposure, all bits are in the high state.

verify/read

To verify correct programming, the EPROM cell can be read using either the verify or read line definitions shown in Table 5, assuming the inhibit bit has not been programmed.

program inhibit

Programming may be inhibited by maintaining a high level input on the $\overline{E}$ pin or $\overline{PGM}$ pin.

read

The EPROM contents may be read independent of the programming cycle, provided the RBIT (ROM protect bit) has not been programmed. The read is accomplished by setting $\overline{E}$ to zero and pulsing $\overline{G}$ low. The contents of the EPROM location selected by the value on the address inputs appear on Q8-Q1.

output disable

During the EPROM programming process, the EPROM data outputs may be disabled, if desired, by establishing the output disable state. This state is selected by setting $\overline{G}$ and $\overline{E}$ pins high. While output disable is selected, Q8-Q1are placed in the high-impedance state.

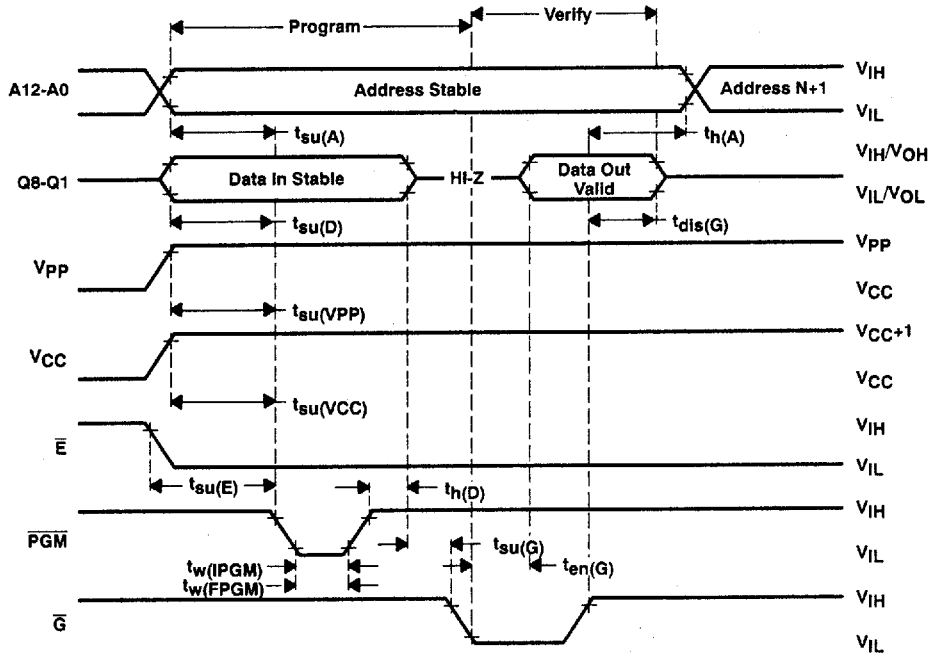
EPROM protection

To protect the proprietary algorithms existing in the code programmed on-chip, the ability to read or verify code from external accesses can be completely disabled. Programming the RBIT disables external access of the EPROM cell and disables the microprocessor mode, making it impossible to access the code resident in the EPROM cell. The only way to remove this protection is to erase the entire EPROM cell, thus removing the proprietary information. The signal requirements for programming this bit are shown in Table 5. The cell can be determined as protected by verifying the programming of the RBIT shown in the table.

standard programming procedure

Before programming, the device must first be completely erased. Then the device can be programmed with the correct code. It is advisable to program unused sections with zeroes as a further security measure. After the programming is complete, the code programmed into the cell should be verified. If the cell passes verification, the next step is to program the ROM protect bit (RBIT). Once the RBIT programming is verified, an opaque label should be placed over the window to protect the EPROM cell from inadvertent erasure by ambient light. At this point, the programming is complete, and the device is ready to be placed into its destination circuit.

program cycle timing



absolute maximum ratings over specified temperature range (unless otherwise noted)†

Supply voltage range, V_{CC} (see Note 6) ..... – 0.3 V to 4.6 V
Input voltage range ..... – 0.3 V to V_{CC} + 0.5
Output voltage range ..... – 0.3 V to V_{CC} + 0.5
Continuous power dissipation ..... 75 mW
Air temperature range above operating devices: L version ..... 0°C to 70°C
A version ..... – 40°C to 85°C
Storage temperature range ..... – 55°C to +150°C

† Stresses beyond those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. This is a stress rating only, and functional operation of the device at these or any other conditions beyond those indicated in the "Recommended Operating Conditions" section of this specification is not implied. Exposure to absolute-maximum-rated conditions for extended periods may affect device reliability.

NOTE 6: All voltage values are with respect to V_{SS}.

recommended operating conditions

			MIN	NOM	MAX	UNIT
V_{CC}	Supply voltage		3.0	3.3	3.6	V
V_{SS}	Supply voltage			0		V
V_{IH}	High-level input voltage	All inputs except CLKIN	2.0			V
		CLKIN	2.5			V
V_{IL}	Low-level input voltage	All inputs			0.55	V
I_{OH}	High-level output current (all outputs)				– 300	μA
I_{OL}	Low-level output current (all outputs)				1.5	mA
T_A	Operating free-air temperature	L version	0		70	°C
		A version	– 40		85	°C

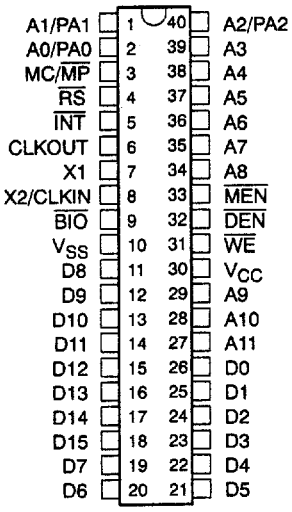
electrical characteristics over specified temperature range (unless otherwise noted)

PARAMETER			TEST CONDITIONS		MIN	TYP†	MAX	UNIT
V_{OH}	High-level output voltage		I_{OH} = MAX		2.0			V
			I_{OH} = 20 μA (see Note 7)		V_{CC} – 0.4‡			V
V_{OL}	Low-level output voltage		I_{OL} = MAX				0.5	V
I_{OZ}	Off-state ouput current		V_{CC} = MAX, V_O = V_{CC}				20	μA
			V_O = V_{SS}				–20	
I_I	Input current		V_I = V_{SS} to V_{CC}	All inputs except CLKIN			±20	μA
			V_I = V_{SS} to V_{CC}	CLKIN			±50	
C_i	Input capacitance	Data bus	f = 1 MHz, All other pins 0 V			25‡		pF
		All others				15‡		
C_o	Output capacitance	Data bus				25‡		pF
		All others				10‡		

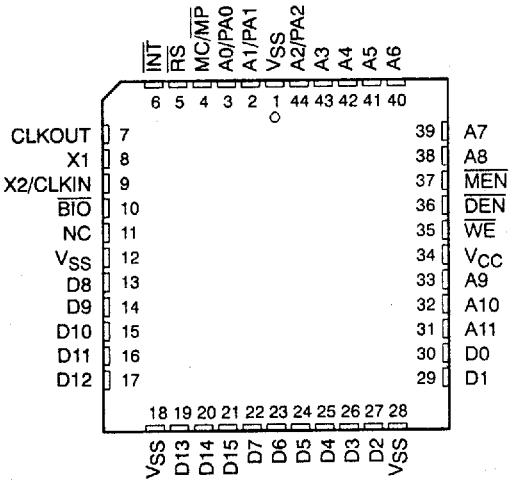
† All typical values are at V_{CC} = 3.3 V, T_A = 25°C.
‡ Values derived from characterization data and not tested.

NOTE 7: This voltage specification is included for interface to HC logic. However, note that all of the other timing parameters defined in this data sheet are specified for TTL logic levels and will differ for HC logic levels.

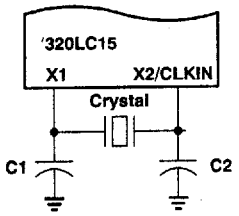
N Package
(Top View)



FN Package
(Top View)



INTERNAL CLOCK OPTION



PARAMETER MEASUREMENT INFORMATION

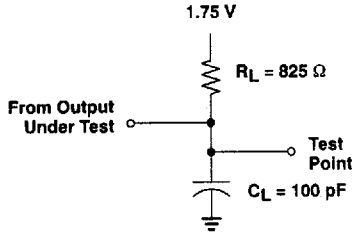


Figure 10. Test Load Circuit

CLOCK CHARACTERISTICS AND TIMING

The 'LC15 can use either its internal oscillator or an external frequency source for a clock.

internal clock option

The internal oscillator is enabled by connecting a crystal across X1 and X2/CLKIN (see Figure 1). The frequency of CLKOUT is one-fourth the crystal fundamental frequency. The crystal should be fundamental mode, and parallel resonant, with an effective series resistance of 30 ohms, a power dissipation of 1 mW, and be specified at a load capacitance of 20 pF.

PARAMETER	TEST CONDITIONS	MIN	NOM	MAX	UNIT
Crystal frequency f_x	T_A = 40°C to 85°C	4.0		16	MHz
C1, C2			10		pF

external clock option

An external frequency source can be used by injecting the frequency directly into X2/CLKIN with X1 left unconnected. The external frequency injected must conform to the specifications listed in the table below.

switching characteristics over recommended operating conditions

PARAMETER		TEST CONDITIONS	MIN	NOM	MAX	UNIT
$t_{c(C)}$	CLKOUT cycle time†	R_L = 825 Ω, C_L = 100 pF, (see Figure 2)	250		1000	ns
$t_{r(C)}$	CLKOUT rise time			10‡		ns
$t_{f(C)}$	CLKOUT fall time			8‡		ns
$t_{w(CL)}$	Pulse duration, CLKOUT low			117‡		ns
$t_{w(CH)}$	Pulse duration, CLKOUT high			115‡		ns
$t_{d(MCC)}$	Delay time, CLKIN↑ to CLKOUT↓		20		70	ns

timing requirements over recommended operating conditions

		MIN	NOM	MAX	UNIT
$t_{c(MC)}$	Master clock cycle time	62.5		150	ns
$t_{r(MC)}$	Rise time, master clock input		5‡	10†	ns
$t_{f(MC)}$	Fall time, master clock input		5‡	10†	ns
$t_{w(MCP)}$	Pulse duration, master clock	0.4 $t_{c(MC)}$‡		0.6 $t_{c(MC)}$‡	ns
$t_{w(MCL)}$	Pulse duration, master clock low at $t_{c(MC)}$ min		26		ns
$t_{w(MCH)}$	Pulse duration, master clock high at $t_{c(MC)}$ min		26		ns

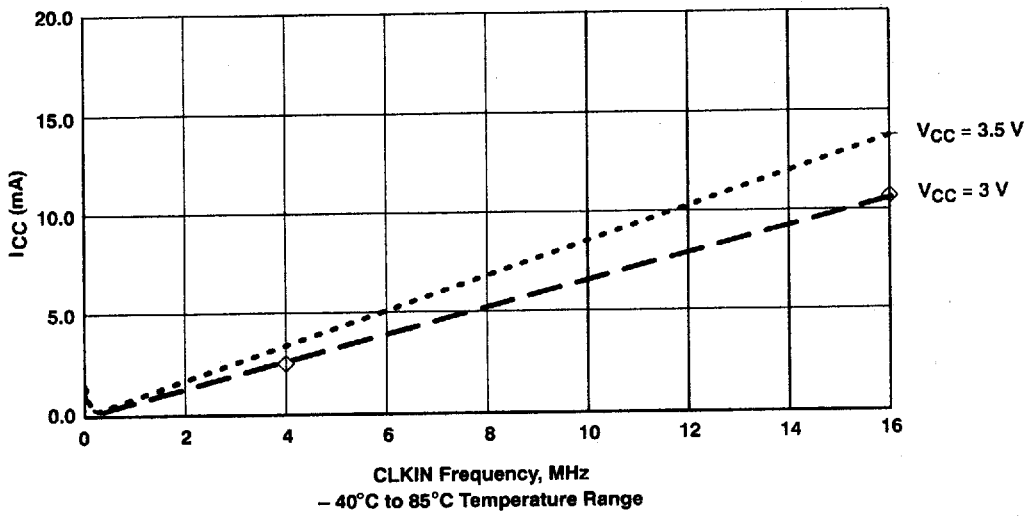
† $t_{c(C)}$ is the cycle time of CLKOUT, i.e., $4t_{c(MC)}$ (4 times CLKIN cycle time if an external oscillator is used)
‡ Values derived from characterization data and not tested.

electrical characteristics over specified temperature range (unless otherwise noted)

PARAMETER	TEST CONDITIONS	MIN	TYP†	MAX	UNIT
I_{CC}‡	f = 16.0 MHz, V_{CC} = 3.6 V, T_A = 0°C to 70°C		15	20	mA

† All typical values are at T_A = 70°C and are used for thermal resistance calculations.
‡ I_{CC} characteristics are inversely proportional to temperature. For I_{CC} dependence on frequency, see figure below.

typical power vs. frequency graph (outputs unloaded)§



§ Device operation is not guaranteed below 4 MHz CLKIN.
Graph is for device in RESET; i.e., only clock-out is driven.

MEMORY AND PERIPHERAL INTERFACE TIMING

switching characteristics over recommended operating conditions

	PARAMETER	TEST CONDITIONS	MIN	MAX	UNIT
t_{d1}	Delay time CLKOUT↓ to address bus valid	R_L = 825Ω, C_L = 100 pF, (see Figure 2)	10†	75	ns
t_{d2}	Delay time CLKOUT↓ to $\overline{MEN}$↓		1/4 $t_{c(C)}$−5†	1/4 $t_{c(C)}$+25	ns
t_{d3}	Delay time CLKOUT↓ to $\overline{MEN}$↑		−10†	30	ns
t_{d4}	Delay time CLKOUT↓ to $\overline{DEN}$↓		1/4 $t_{c(C)}$−5†	1/4 $t_{c(C)}$+25	ns
t_{d5}	Delay time CLKOUT↓ to $\overline{DEN}$↑		−10†	30	ns
t_{d6}	Delay time CLKOUT↓ to $\overline{WE}$↓		1/2 $t_{c(C)}$−5†	1/2 $t_{c(C)}$+25	ns
t_{d7}	Delay time CLKOUT↓ to $\overline{WE}$↑		−10†	30	ns
t_{d8}	Delay time CLKOUT↓ to data bus OUT valid			1/4 $t_{c(C)}$+75	ns
t_{d9}	Time after CLKOUT↓ that data bus starts to be driven		1/4 $t_{c(C)}$−5†		ns
t_{d10}	Time after CLKOUT↓ that data bus stops being driven			1/4 $t_{c(C)}$+60	ns
t_v	Data bus OUT valid after CLKOUT↓		1/4 $t_{c(C)}$−10		ns
$t_{h(A\text{-}WMD)}$	Address hold time after WE↑, $\overline{MEN}$↑, or $\overline{DEN}$↑ (see Note 14)		0†		ns
$t_{su(A\text{-}MD)}$	Address bus setup time to $\overline{DEN}$↓		−4†		ns

† Values derived from characterization data and not tested.
NOTE 14: Address bus will be valid upon $\overline{WE}$↑, $\overline{MEN}$↑, or $\overline{DEN}$↑.

timing requirements over recommended operating conditions

		TEST CONDITIONS	MIN	NOM	MAX	UNIT
$t_{su(D)}$	Setup time data bus valid prior to CLKOUT↓	R_L = 825Ω, C_L = 100 pF, (see Figure 2)	56			ns
$t_{h(D)}$	Hold time, data bus held valid after CLKOUT↓ (see Note 9)		0			ns

NOTE 9: Data may be removed from the data bus upon $\overline{MEN}$↑ or $\overline{DEN}$↑ preceding CLKOUT↓.

RESET ($\overline{RS}$) TIMING

switching characteristics over recommended operating conditions

		TEST CONDITIONS	MIN	NOM	MAX	UNIT
t_{d11}	Delay time, $\overline{DEN}$↑, $\overline{WE}$↑, and $\overline{MEN}$↑ from $\overline{RS}$	R_L = 825Ω, C_L = 100 pF, (see Figure 2)			$1/2t_{c(C)}+75$	ns
$t_{dis(R)}$	Data bus disable time after $\overline{RS}$				$1/4t_{c(C)}+75$	ns

† These parameters do not apply to this device.

timing requirements over recommended operating conditions

		MIN	NOM	MAX	UNIT
$t_{su(R)}$	Reset ($\overline{RS}$) setup time prior to CLKOUT (see Note 10)	85			ns
$t_{w(R)}$	$\overline{RS}$ pulse duration	$5t_{c(C)}$			ns

NOTE 10: $\overline{RS}$ can occur anytime during a clock cycle. Time given is minimum to ensure synchronous operation.

INTERRUPT ($\overline{INT}$) TIMING

timing requirements over recommended operating conditions

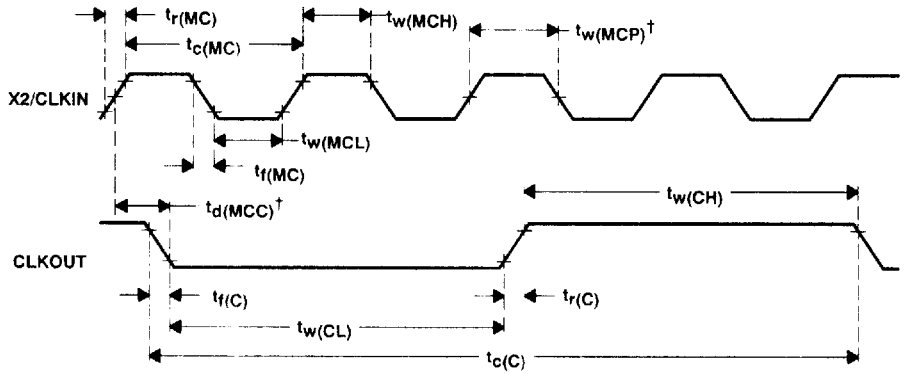
		MIN	NOM	MAX	UNIT
$t_{F(INT)}$	Fall time, $\overline{INT}$			15	ns
$t_{w(INT)}$	Pulse duration, $\overline{INT}$	$t_{c(C)}$			ns
$t_{su(INT)}$	Setup time, $\overline{INT}$↓ before CLKOUT↓	85			ns

I/O ($\overline{BIO}$) TIMING

timing requirements over recommended operating conditions

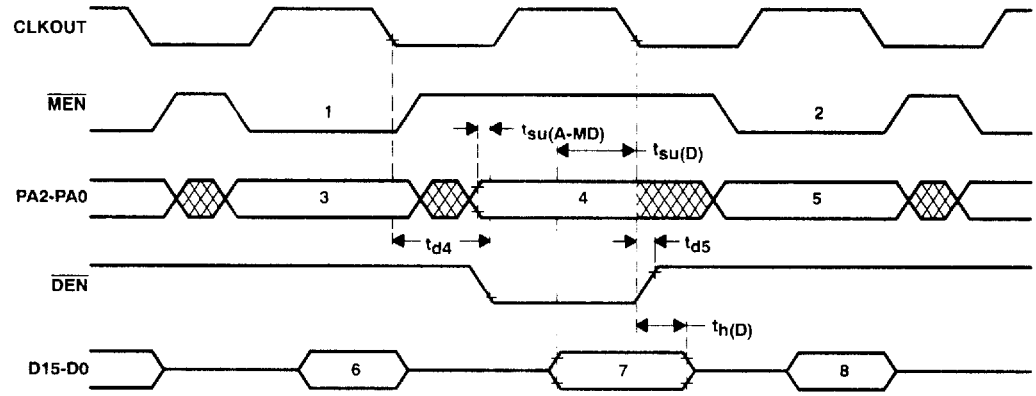
		MIN	NOM	MAX	UNIT
$t_{f(IO)}$	Fall time $\overline{BIO}$			15	ns
$t_{w(IO)}$	Pulse duration $\overline{BIO}$	$t_{c(C)}$			ns
$t_{su(IO)}$	Setup time $\overline{BIO}$↓ before CLKOUT↓	85			ns

clock timing



† $t_{d(MCC)}$ and $t_{w(MCP)}$ are referenced to an intermediate level of 1.5 V on the CLKIN waveform.

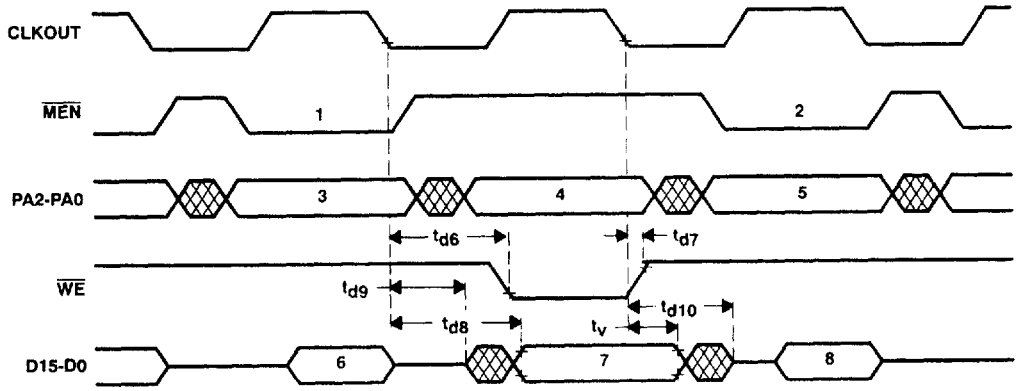
IN instruction timing



Legend:

1. IN Instruction Prefetch
2. Next Instruction Prefetch
3. Address Bus Valid
4. Peripheral Address Valid
5. Address Bus Valid
6. Instruction Valid
7. Data Input Valid
8. Instruction Valid

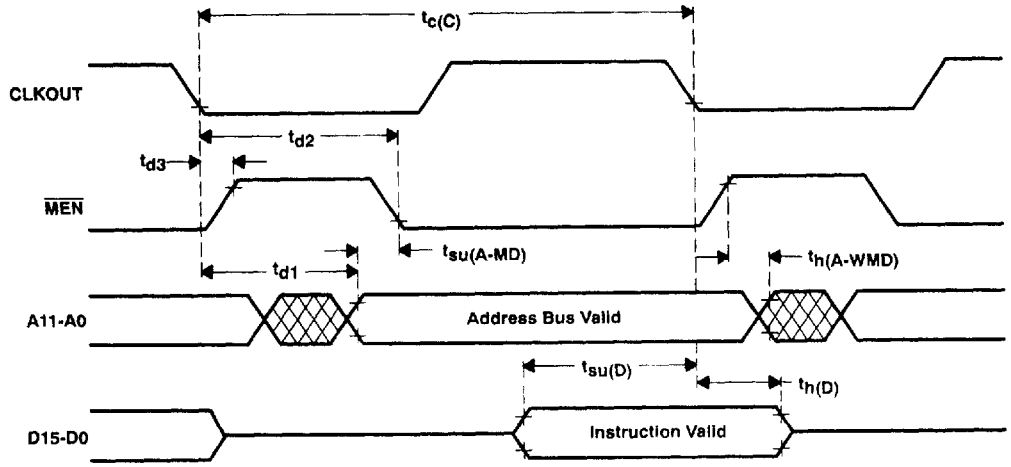
OUT instruction timing



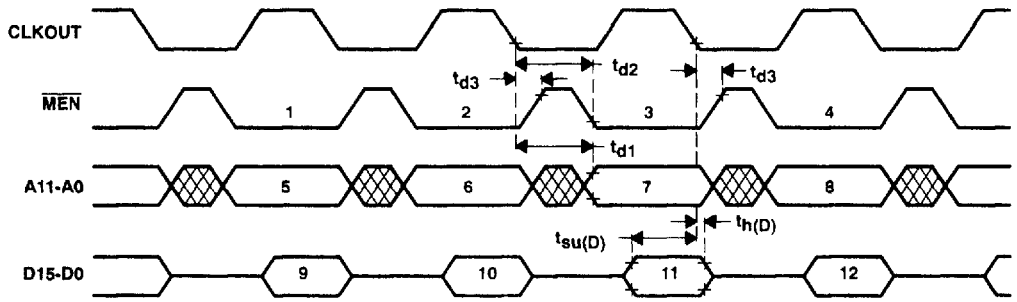
Legend:

1. OUT Instruction Prefetch
2. Next Instruction Prefetch
3. Address Bus Valid
4. Peripheral Address Valid
5. Address Bus Valid
6. Instruction Valid
7. Data Output Valid
8. Instruction Valid

external memory read timing



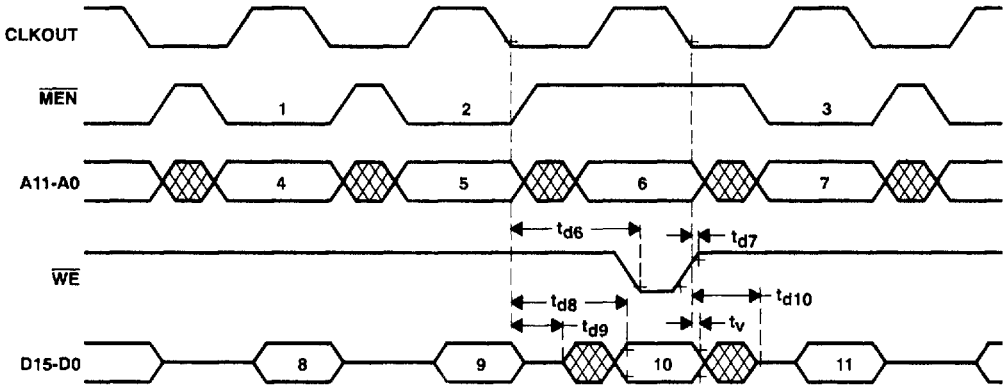
TBLR Instruction timing



Legend:

1. TBLR Instruction Prefetch
2. Dummy Prefetch
3. Data Fetch
4. Next Instruction Prefetch
5. Address Bus Valid
6. Address Bus Valid
7. Address Bus Valid
8. Address Bus Valid
9. Instruction Valid
10. Instruction Valid
11. Data Input Valid
12. Instruction Valid

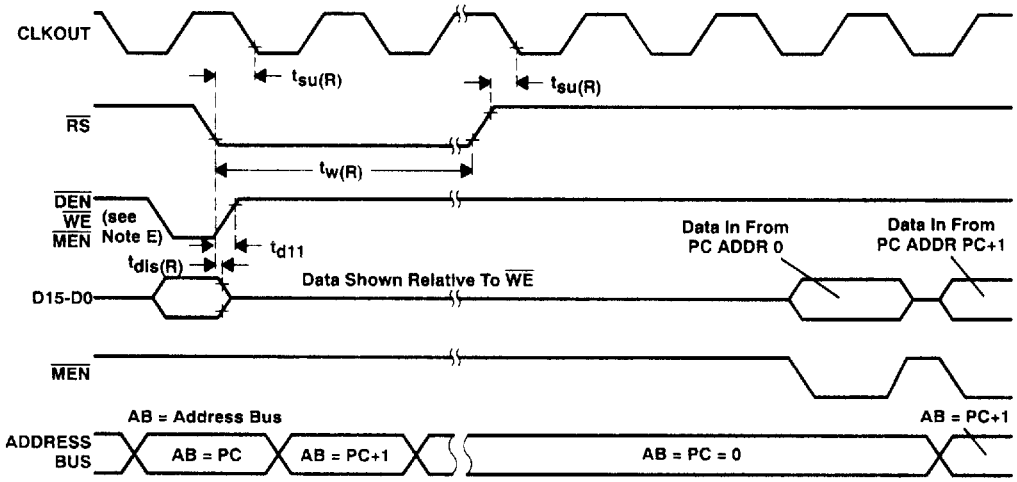
TBLW Instruction timing



Legend:

1. TBLW Instruction Prefetch
2. Dummy Prefetch
3. Next Instruction Prefetch
4. Address Bus Valid
5. Address Bus Valid
6. Address Bus Valid
7. Address Bus Valid
8. Instruction Valid
9. Instruction Valid
10. Data Output Valid
11. Instruction Valid

reset timing



NOTES: A. $\overline{RS}$ forces $\overline{DEN}$, $\overline{WE}$, and $\overline{MEN}$ high and places data bus D0 through D15 in a high-impedance state. AB outputs (and program counter) are synchronously cleared to zero after the next complete CLK cycle from $\overline{RS}\downarrow$.

B. $\overline{RS}$ must be maintained for a minimum of five clock cycles.

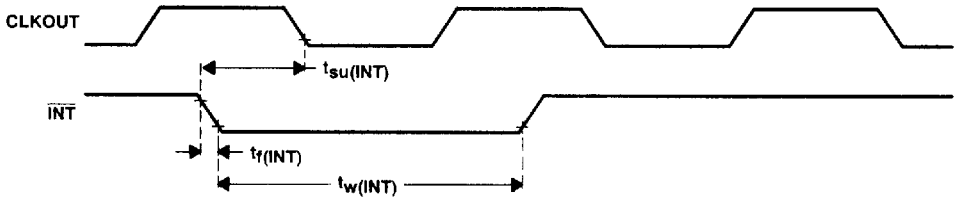
C. Resumption of normal program will commence after one complete CLK cycle from $\overline{RS}\uparrow$.

D. Due to the synchronization action on $\overline{RS}$, time to execute the function can vary dependent upon when $\overline{RS}\uparrow$ or $\overline{RS}\downarrow$ occur in the CLK cycle.

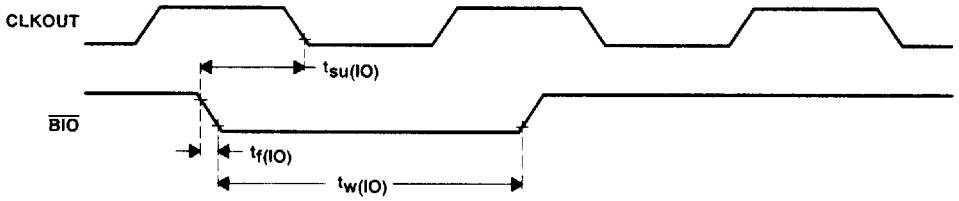
E. Diagram shown is for definition purpose only. $\overline{DEN}$, $\overline{WE}$, and $\overline{MEN}$ are mutually exclusive.

F. During a write cycle, $\overline{RS}$ may produce an invalid write address.

interrupt timing



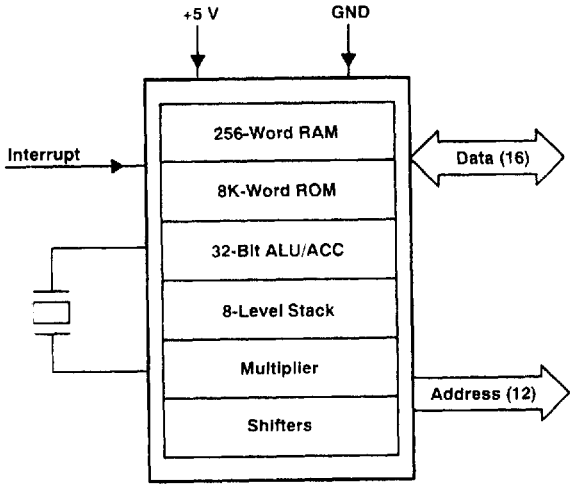
$\overline{BIO}$ timing



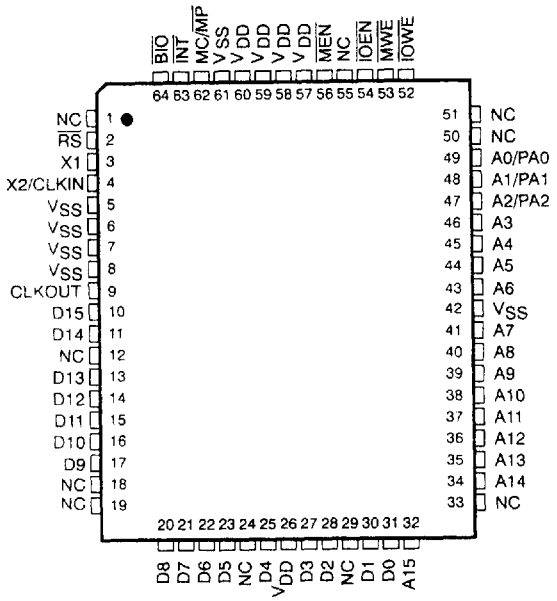
JANUARY 1987 — REVISED JULY 1991

Key Features: TMS320C16

- 114-ns Instruction Cycle Time
- 256 Words of On-Chip Data RAM
- 8K Words of On-Chip Program ROM
- 64K Words Total External Memory at Full Speed
- 8 Level Stack
- 32-Bit ALU/Accumulator
- 16 × 16-Bit Multiplier With 32-Bit Product
- 16-Bit Barrel Shifter
- Eight Input and Eight Output Channels
- Simple Memory and I/O Interface:
  - — Memory Write Enable Signal $\overline{\text{MWE}}$
  - — I/O Write Enable Signal $\overline{\text{IOWE}}$
- Single 5-V Supply
- 64-Pin Quad Flatpack (PG Suffix)
- Operating Free-Air Temperature Range
  . . . 0°C to 70°C



PG Package
(Top View)



TERMINAL FUNCTIONS

PIN NAME	PIN NO.	I/O/Z†	DESCRIPTION
			ADDRESS/DATA BUSES
A15 MSB	32	I/O/Z	Program memory address bus A15 (MSB) through A0 (LSB) and port addresses PA2 (MSB) through PA0 (LSB). Addresses A15 through A0 are always active and never go to high impedance. During execution of the IN and OUT instructions, pins A2 through A0 carry the port addresses. (Address pins A15 through A3 are always driven low on IN and OUT instruction.
A14	34		
A13	35		
A12	36		
A11	37		
A10	38		
A9	39		
A8	40		
A7	41		
A6	43		
A5	44		
A4	45		
A3	46		
A2/PA2	47		
A1/PA1	48		
A0/PA0	49		
D15 MSB	10	I/O/Z	Parallel data bus D15 (MSB) through D0 (LSB). The data bus is always in the high-impedance state except when $\overline{\text{IOWE}}$ or $\overline{\text{MWE}}$ are active (low).
D14	11		
D13	13		
D12	14		
D11	15		
D10	16		
D9	17		
D8	20		
D7	21		
D6	22		
D5	23		
D4	25		
D3	27		
D2	28		
D1	30		
D0 LSB	31		

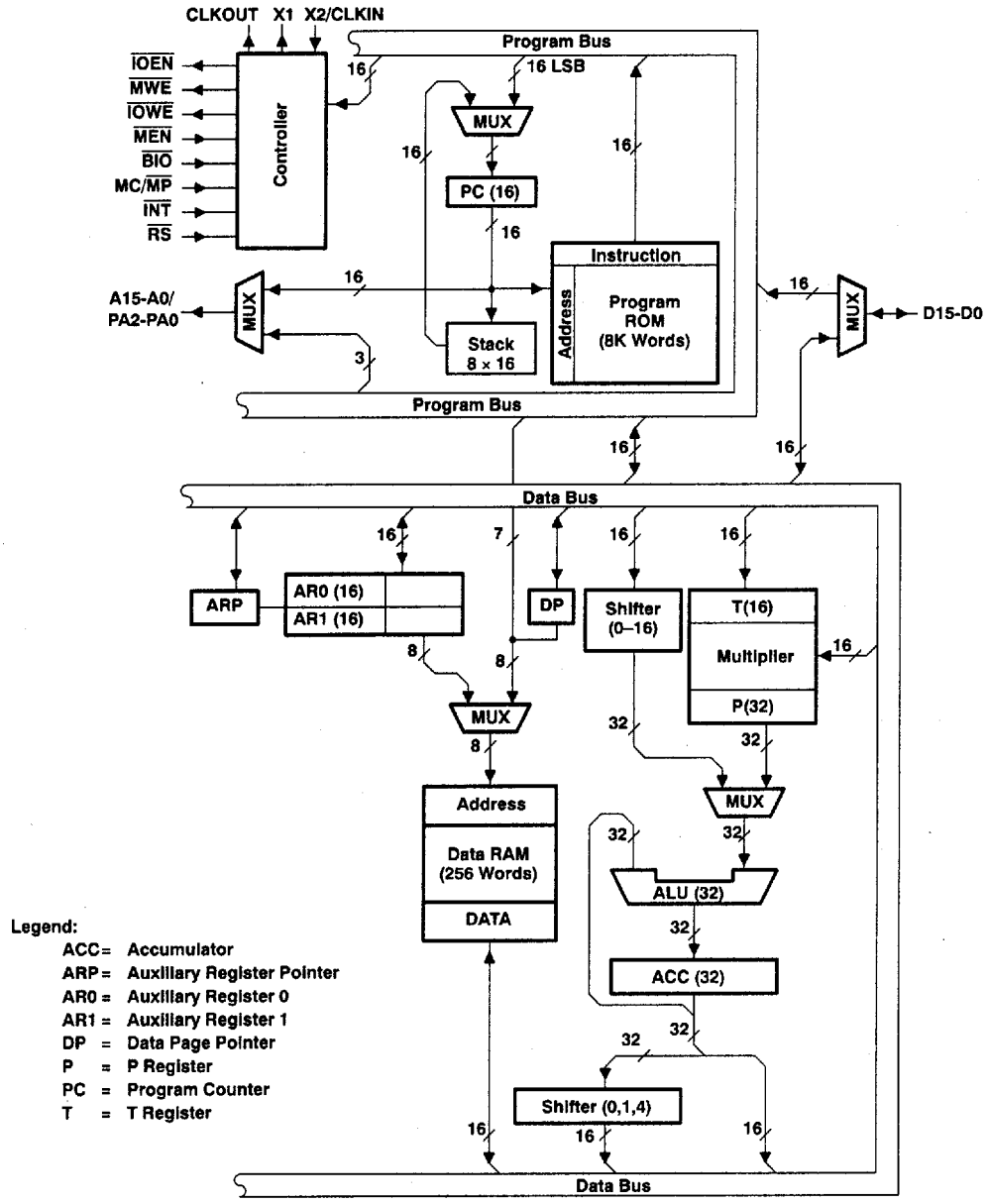
† Input/Output/High-impedance state.

TERMINAL FUNCTIONS (concluded)

PIN NAME	PIN NO.	I/O/Z†	DESCRIPTION
			INTERRUPT AND MISCELLANEOUS SIGNALS
$\overline{BIO}$	64	I	External polling input. Polled by BIOZ instruction. If low, the device branches to the address specified by the instruction.
$\overline{IOEN}$	54	O	Data enable for device input data. When active (low), $\overline{IOEN}$ indicates that the device will accept data from the data bus. $\overline{IOEN}$ is active only during the IN instruction. When $\overline{IOEN}$ is active, $\overline{MEN}$, $\overline{IOWE}$, and $\overline{MWE}$ will always be inactive (high).
$\overline{IOWE}$	52	O	Write enable for device output data. When active (low), $\overline{IOWE}$ indicates that data will be output from the device on the data bus. $\overline{IOWE}$ is active only during the OUT instruction. When $\overline{IOWE}$ is active, $\overline{MEN}$, $\overline{IOEN}$, and $\overline{MWE}$ will always be inactive (high).
$\overline{INT}$	63	I	External interrupt input. The interrupt signal is generated by applying a negative-going edge to the $\overline{INT}$ pin. The edge is used to latch the interrupt flag register (INTF) until an interrupt is granted by the device. An active low level will also be sensed.
MC/$\overline{MP}$	62	I	Memory mode select pin. High selects the microcomputer mode, in which 8K words of on-chip program memory are available. A low on MC/$\overline{MP}$ pin enables the microprocessor mode. In this mode, the entire memory space is external; i.e., addresses 0 through 65535.
$\overline{MEN}$	56	O	Memory enable. $\overline{MEN}$ is an active (low) control signal generated by the device to enable instruction fetches from program memory. $\overline{MEN}$ will be active on instructions fetched from both internal and external memory. When $\overline{MEN}$ is active, $\overline{MWE}$, $\overline{IOWE}$, and $\overline{IOEN}$ will be inactive (high).
$\overline{MWE}$	53	O	Write enable for device output data. When active (low), $\overline{MWE}$ indicates that data will be output from the device on the data bus. $\overline{MWE}$ is active only during the TBLW instruction. When $\overline{MWE}$ is active, $\overline{MEN}$, $\overline{IOEN}$, and $\overline{IOWE}$ will always be inactive (high).
NC	1, 12, 18, 19, 24, 29, 33, 50, 51, 55	—	No connection.
$\overline{RS}$	2	I	Schmitt-triggered input for initializing the device. When held active for a minimum of five clock cycles. $\overline{IOEN}$, $\overline{IOWE}$, $\overline{MWE}$, and $\overline{MEN}$ are forced high; and, the data bus (D15 through D0) is not driven. The program counter (PC) and the address bus (A15 through A0) are then synchronously cleared after the next complete clock cycle from the falling edge of $\overline{RS}$. Reset also disables the interrupt, clears the interrupt flag register, and leaves the overflow mode register unchanged. The device can be held in the reset state indefinitely.
			SUPPLY/OSCILLATOR SIGNALS
PIN NAME	**PIN NO.**	**I/O/Z†**	**DESCRIPTION**
CLKOUT	9	O	System clock output (one-fourth crystal/CLKIN frequency).
V_{DD}	26, 57, 58, 59, 60	I	5-V suppy pins.
V_{SS}	5, 6, 7, 8, 42, 61	I	Ground pins.
X1	3	O	Crystal output pin for internal oscillator. If the internal oscillator is not used, this pin should be left unconnected.
X2/CLKIN	4	I	Input pin to the internal oscillator (X2) from the crystal. Alternatively, an input pin for an external oscillator (CLKIN).

† Input/Output/High-impedance state.

functional block diagram



Legend:
- ACC = Accumulator
- ARP = Auxiliary Register Pointer
- AR0 = Auxiliary Register 0
- AR1 = Auxiliary Register 1
- DP = Data Page Pointer
- P = P Register
- PC = Program Counter
- T = T Register

absolute maximum ratings over operating free-air temperature range (unless otherwise noted)†

Supply voltage range, V_{CC} (see Note 6) ..... – 0.3 V to 7 V
Input voltage range ..... – 0.3 V to 7 V
Output voltage range ..... – 0.3 V to 7 V
Continuous power dissipation ..... 0.5 W
Operating free-air temperature: ..... 0°C to 70°C
Storage temperature ..... – 55 °C to 150 °C

† Stresses beyond those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. This is a stress rating only, and functional operation of the device at these or any other conditions beyond those indicated in the "Recommended Operating Conditions" section of this specification is not implied. Exposure to absolute-maximum-rated conditions for extended periods may affect device reliability.

NOTE 6: All voltage values are with respect to V_{SS}.

recommended operating conditions

			MIN	NOM	MAX	UNIT
V_{CC}	Supply voltage		4.75	5	5.25	V
V_{SS}	Supply voltage			0		V
V_{IH}	High-level input voltage	All inputs except CLKIN	2			V
		CLKIN	3			V
V_{IL}	Low-level input voltage	All inputs except MC/$\overline{MP}$			0.8	V
		MC/$\overline{MP}$			0.6	V
I_{OH}	High-level output current, all outputs				–300	μA
I_{OL}	Low-level output current				2	mA
T_A	Operating free-air temperature		0		70	°C

electrical characteristics over recommended operating free-air temperature range (unless otherwise noted)

PARAMETER			TEST CONDITIONS		MIN	TYP	MAX	UNIT
V_{OH}	High-level output voltage		I_{OH} = MAX		2.4	3		V
			I_{OH} = 20 μA		V_{CC} – 0.4			
V_{OL}	Low-level output voltage		I_{OL} = MAX			0.3	0.5	V
I_{OZ}	Off-state output current		V_{CC} = MAX	V_O = 2.4 V			20	μA
				V_O = 0.4 V			– 20	
I_I	Input current		V_{CC} = V_{SS} to V_{CC}	All inputs except CLKIN			±20	μA
				CLKIN			±50	
I_{CC}	Supply current		f = 35 MHz, V_{CC} = 5.25 V			60	75	mA
C_i	Input capacitance	Data bus	f = 1 MHz, all other pins 0 V			25		pF
		All others				15		
C_o	Output capacitance	Data bus				25		pF
		All others				10		

internal clock option

PARAMETER	TEST CONDITIONS	MIN	NOM	MAX	UNIT
Crystal frequency, f_x	T_A = 0°C to 70°C	6.7		35.1	MHz
C1, C2	T_A = 0°C to 70°C		10		pF

timing requirements over recommended operating conditions

		MIN	NOM	MAX	UNIT
$t_{c(MC)}$	Master clock cycle time	28.49	28.57	150	ns
$t_{r(MC)}$	Rise time, master clock input		5	10	ns
$t_{f(MC)}$	Fall time, master clock input		5	10	ns
$t_{w(MCP)}$	Pulse duration, master clock	$0.45t_{c(C)}$		$0.55t_{c(C)}$	ns
$t_{w(MCL)}$	Pulse duration, master clock low		10		ns
$t_{w(MCH)}$	Pulse duration, master clock high		10		ns

switching characteristics over recommended operating conditions

PARAMETER		MIN	NOM	MAX	UNIT
$t_{c(C)}$	CLKOUT cycle time	113.96	114.3	600	ns
$t_{r(C)}$	CLKOUT rise time		10		ns
$t_{f(C)}$	CLKOUT fall time		8		ns
$t_{w(CL)}$	Pulse duration, CLKOUT low		49		ns
$t_{w(CH)}$	Pulse duration, CLKOUT high		47		ns
$t_{d(MCC)}$	Delay time, CLKIN↑ to CLKOUT↓	5		50	ns

MEMORY AND PERIPHERAL INTERFACE TIMING

switching characteristics over recommended operating conditions

	PARAMETER	MIN	NOM	MAX	UNIT
t_{d1}	Delay time, $\overline{MEN}\uparrow$, $\overline{MWE}\uparrow$, $\overline{IOEN}\uparrow$, $\overline{IOWE}\uparrow$, to next address bus valid	0		35	ns
t_{d2}	Delay time, CLKOUT↓ to $\overline{MEN}\downarrow$	$^1/_4t_{c(C)} - 5$		$^1/_4t_{c(C)} + 12$	ns
t_{d3}	Delay time, CLKOUT↓ to $\overline{MEN}\uparrow$	−3		6	ns
t_{d4}	Delay time, CLKOUT↓ to $\overline{IOEN}\downarrow$	$^1/_4t_{c(C)} - 5$		$^1/_4t_{c(C)} + 12$	ns
t_{d5}	Delay time, CLKOUT↓ to $\overline{IOEN}\uparrow$	−3		6	ns
t_{d6}	Delay time, CLKOUT↓ to $\overline{MWE}\downarrow$, $\overline{IOWE}\downarrow$	$^1/_2t_{c(C)} - 5$		$^1/_2t_{c(C)} + 12$	ns
t_{d7}	Delay time, CLKOUT↓ to $\overline{MWE}\uparrow$, $\overline{IOWE}\uparrow$	−3		6	ns
t_{d8}	Delay time, $\overline{MWE}\downarrow$, $\overline{IOwE}\downarrow$, data bus out valid			0	ns
$t_{d9(CLK)}$	Delay time, CLKOUT↓ to data bus starts to be driven	$^1/_4t_{c(C)} - 5$			ns
$t_{d9(MEN)}$	Delay time, $\overline{MEN}\uparrow$, to data bus starts to be driven	$^1/_4t_{c(C)}$			ns
$t_{d10(CLK)}$	Delay time, CLKOUT↓ to data bus stops being driven			15	ns
$t_{d10(WE)}$	Delay time, $\overline{MWE}\uparrow$, $\overline{IOWE}\uparrow$, data bus stops being driven			20	ns
t_v	Data bus OUT valid after $\overline{MWE}\uparrow$, $\overline{IOWE}\uparrow$	5	10		ns
$t_{h(A\text{-}WMD)}$	Address bus hold time after $\overline{MWE}\uparrow$, $\overline{MEN}\uparrow$, $\overline{IOWE}\uparrow$, or $\overline{IOEN}\uparrow$	0	2		ns
$t_{su(A\text{-}MD)}$	Address bus setup time prior to $\overline{MEN}\downarrow$, $\overline{IOEN}\downarrow$	5			ns

timing requirements over recommended operating conditions

		MIN	MAX	UNIT
$t_{su(D)}$	Setup time, data bus valid prior to $\overline{MEN}\uparrow$, $\overline{IOEN}\uparrow$	35		ns
$t_{h(D)}$	Hold time, data bus held valid after $\overline{MEN}\uparrow$, $\overline{IOEN}\uparrow$	0		ns

RESET ($\overline{RS}$) TIMING

switching characteristics over recommended operating conditions

	PARAMETER	MIN	MAX	UNIT
t_{d11}	Delay time, $\overline{IOEN}\uparrow$, $\overline{IOWE}\uparrow$, $\overline{MWE}\uparrow$, and $\overline{MEN}\uparrow$ from $\overline{RS}$		$^1/_2t_{c(C)} + 50$	ns
$t_{dis(R)}$	Data bus disable time after $\overline{RS}$		$^1/_4t_{c(C)} + 50$	ns

timing requirements over recommended operating conditions

		MIN	MAX	UNIT
$t_{su(R)}$	Reset ($\overline{RS}$) setup time prior to CLKOUT	30		ns
$t_{w(R)}$	$\overline{RS}$ pulse duration	$5t_{c(C)}$		ns

INTERRUPT ($\overline{INT}$) TIMING

timing requirements over recommended operating conditions

		MIN	MAX	UNIT
$t_{f(INT)}$	Fall time, $\overline{INT}$		15	ns
$t_{w(INT)}$	Pulse duration, $\overline{INT}$	$t_{c(C)}$		ns
$t_{su(INT)}$	Setup time, $\overline{INT}\downarrow$ before CLKOUT$\downarrow$	30		ns

IO ($\overline{BIO}$) TIMING

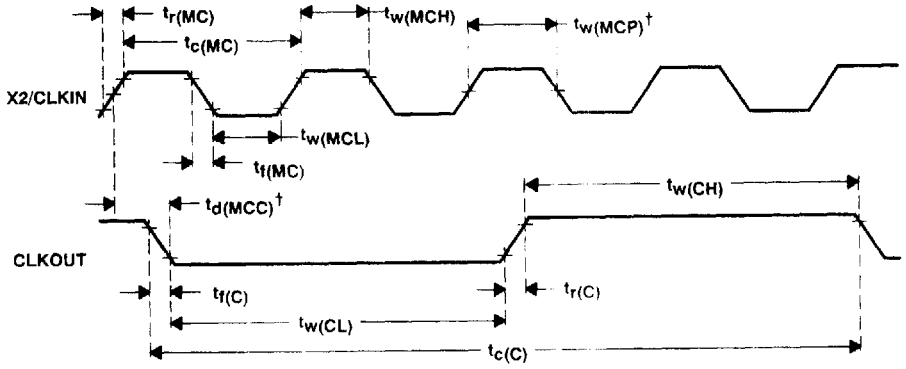
timing requirements over recommended operating conditions

		MIN	MAX	UNIT
$t_{f(IO)}$	Fall time, $\overline{BIO}$		15	ns
$t_{w(IO)}$	Pulse duration, $\overline{BIO}$	$t_{c(C)}$		ns
$t_{su(IO)}$	Setup time, $\overline{BIO}\downarrow$ before CLKOUT$\downarrow$	30		ns

TIMING DIAGRAMS

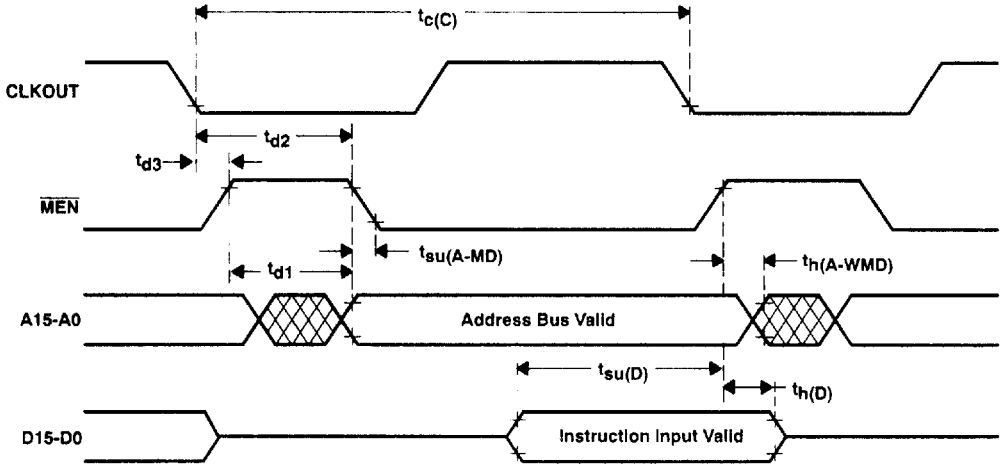
Timing measurements are referenced to and from a low voltage of 0.8 volts and a high voltage of 2.0 volts, unless otherwise noted.

clock timing

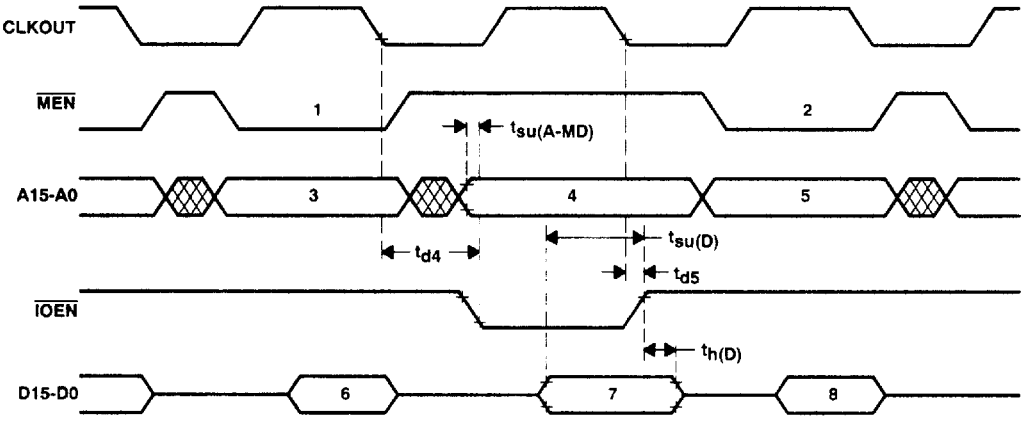


† $t_{d(MCC)}$ and $t_{w(MCP)}$ are referenced to an intermediate level of 1.5 V on the CLKIN waveform.

memory read timing



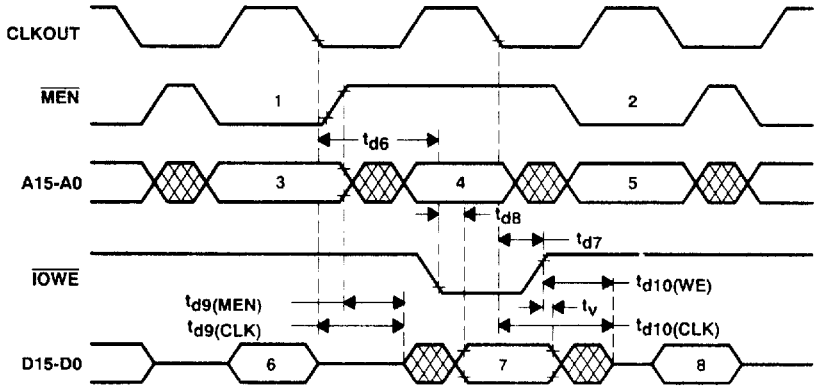
IN instruction timing



Legend:

1. IN instruction prefetch
2. Next instruction prefetch
3. Address bus valid
4. Peripheral address valid
5. Address bus valid
6. Instruction input valid
7. Data input valid
8. Instruction input valid

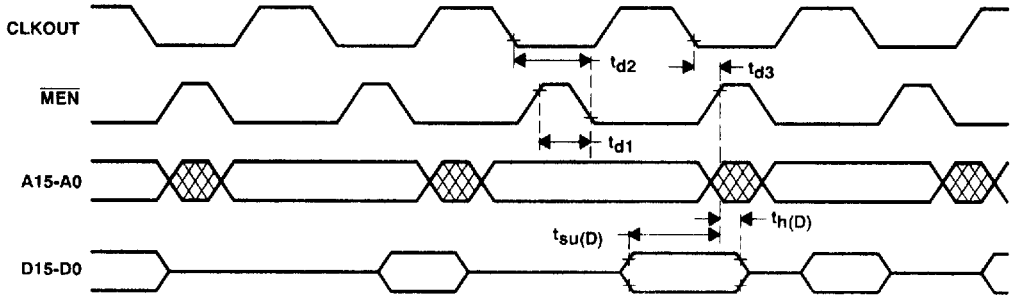
OUT instruction timing



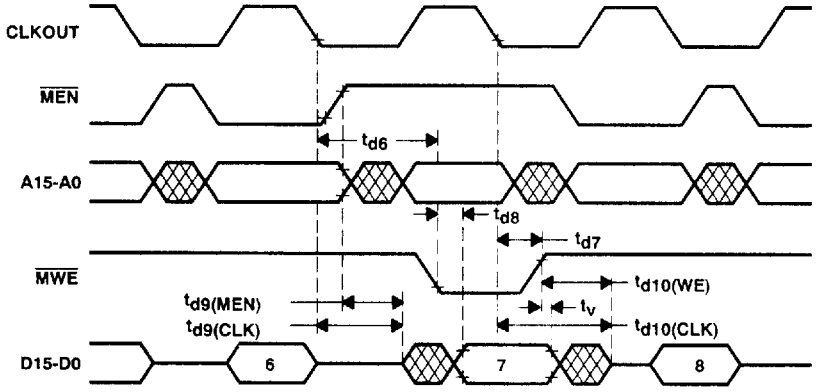
Legend:

1. OUT instruction prefetch
2. Next instruction prefetch
3. Address bus valid
4. Peripheral address valid
5. Address bus valid
6. Instruction valid
7. Data output valid
8. Instruction valid

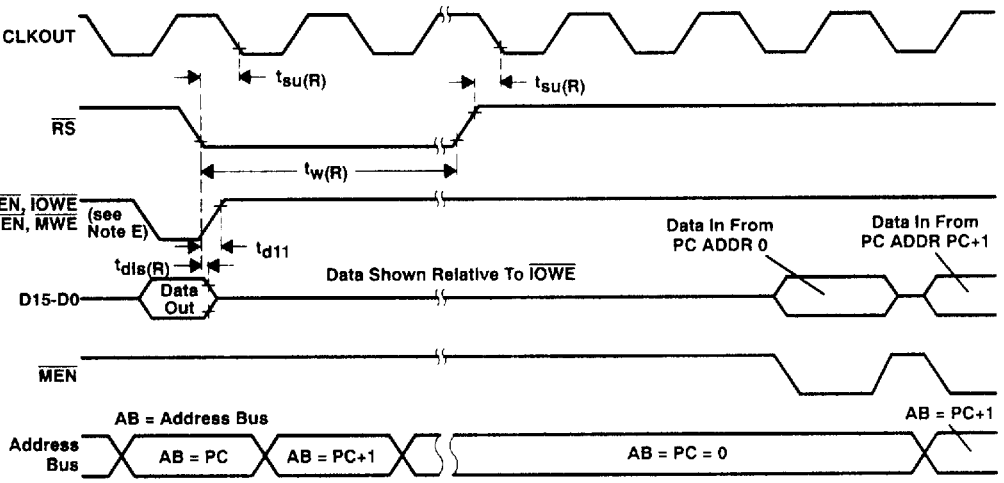
TBLR instruction timing



TBLW instruction timing



reset timing



NOTES: A. $\overline{RS}$ forces $\overline{IOEN}$, $\overline{IOWE}$, $\overline{MWE}$, and $\overline{MEN}$ high and places data bus D0 through D15 in a high-impedance state. AB outputs (and program counter) are synchronously cleared to zero after the next complete CLK cycle from $\overline{RS}\downarrow$.

B. $\overline{RS}$ must be maintained for a minimum of five clock cycles

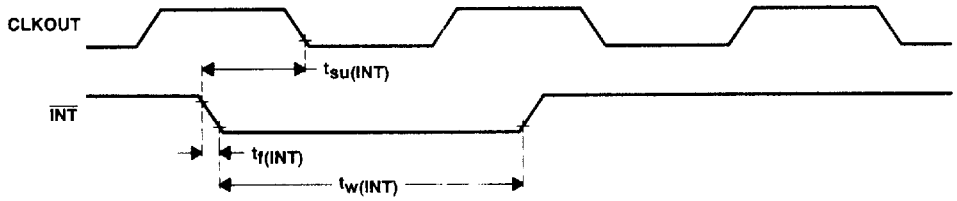
C. Resumption of normal program will commence after one complete CLK cycle from $\overline{RS}\uparrow$.

D. Due to the synchronization action on $\overline{RS}$, time to execute the function can vary dependent upon when $\overline{RS}\uparrow$ or $\overline{RS}\downarrow$ occur in the CLK cycle.

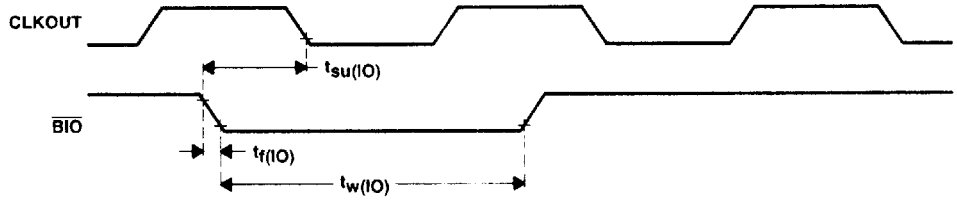
E. Diagram shown is for definition purpose only. $\overline{IOEN}$, $\overline{IOWE}$ $\overline{MWE}$, and $\overline{MEN}$ are mutually exclusive.

F. During a write cycle, $\overline{RS}$ may produce an invalid write address.

interrupt timing



$\overline{BIO}$ timing



design considerations for interfacing to SRAM, EPROM and peripherals

The 'C16 differs somewhat from the other members of the 'C1x family of digital signal processors (DSPs). Additional control signals are available for easier interface to external memory or peripherals, and the memory write cycle timings have been changed.

The discussion here will center around changes in t_v and its impact upon SRAM, EPROM and peripherals/latches interfaces

Access time requirements for interface may be defined relative to

1. Valid address (t_a);
2. $\overline{MEN}/\overline{IOEN}$, [($t_{a(MEN)}$)];

Figure 11 and the following examples summarize these timings at 35 MHz CLKIN.

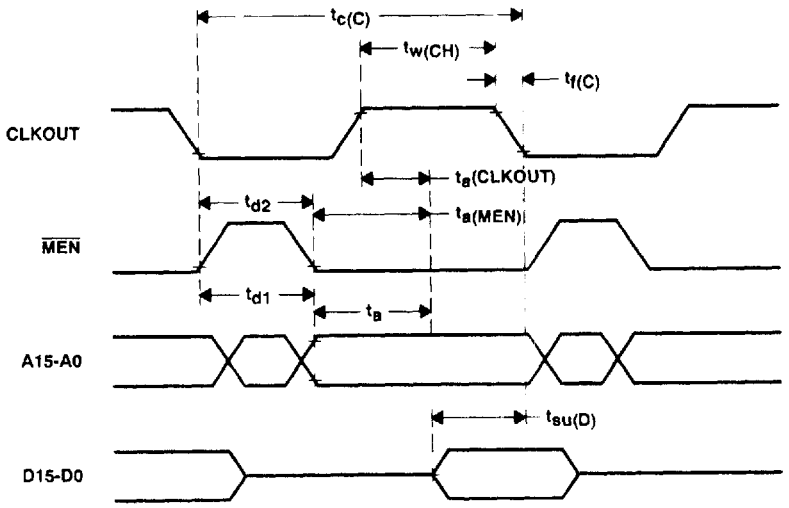


Figure 11.

where:

t_a : (access time from address valid) = $t_{c(C)} - t_{d1} - t_{su(D)}$ = 44.3 ns
$t_{a(MEN)}$: (access time from $\overline{MEN}$ valid) = $t_{c(C)} - t_{d2} - t_{su(D)} + t_{d3}$ = 35.73 ns

and where (for 35 MHz CLKIN):

$t_{c(C)}$ = 114.3 ns
t_{d1} = 35 ns
t_{d2} = [1/4 × (114.3) + 12] ns
$t_{su(D)}$ = 35 ns
$t_{w(CH)}$ = 47 ns nominal
$t_{f(C)}$ = 8 ns nominal

In addition to the above timings, t_v must be taken into account. t_v is the time that the data bus is guaranteed to be held after the rising edge of $\overline{MWE}$ or $\overline{IOWE}$. In other 'C1x devices, the value of t_v was referenced to CLKOUT↓ and not $\overline{WE}$↑ (see Figure 12). For the 'C16, t_v is a minimum of 5 ns. This implies that $\overline{MWE}$ and $\overline{IOWE}$ must be tied directly to the external device. If required, decode logic must be added to an input other than the read/write input — for example, the chip select on SRAMs. If the external device does not have two inputs, then transparent latches must be added to extend the time data is held on the data bus. These latches must be off the bus prior to the next instruction (see Figure 12).

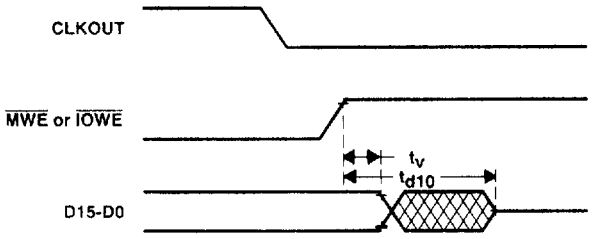


Figure 12.

where:

t_v = 5 ns (min)
t_{d10} = 15 ns (max)

There is a potential for bus conflict on the prefetch and execution of a TBLW or an OUT instruction. Figure 13 details the timings to be considered. In addition to the timings for the 'C16, timing definitions for interface are also included.

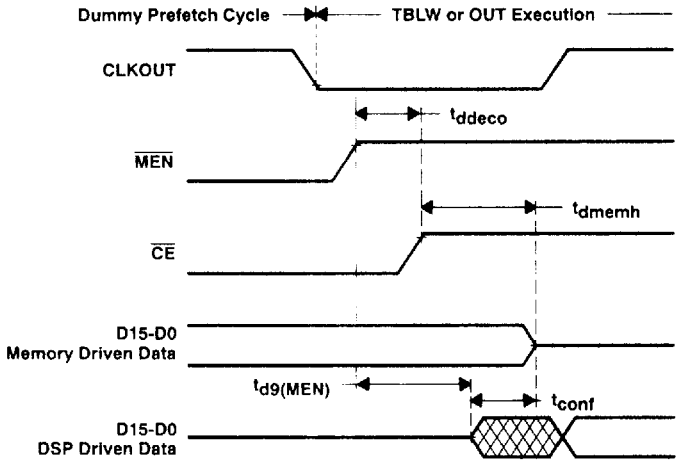


Figure 13.

where:

t_{conf} (data bus conflict time) = $t_{ddeco} + t_{dmemh} - t_{d9(MEN)}$

with:

t_{ddeco} : decode delay time to make the CE or OE signal
t_{dmemh} : memory data hold time from CE or OE
t_{d9} : delay time, $\overline{MEN}$ to data bus starts being driven
t_{d9} : (at 35 MHz CLKIN) = $[1/4t_{c(C)}] = [1/4(114.3)] = 28.58$ ns

If t_{conf} is less than or equal to zero, data bus conflict does not occur.
If t_{conf} is greater than zero, data conflict occurs.

Note that the following discussion is for CLKIN of 35 MHz.

static memory with output enable and write enable/chip select

The following SRAMs are able to interface directly to the 'C16, needing only to directly connect the 'C16 memory control signals $\overline{MEN}$ and $\overline{MWE}$ to the memory. Device select decode is accomplished with address decode and then input to the device chip select.

PRODUCT	t_{ddeco}	t_{dmemh}	t_{dconf}	UNITS
TC55645-35	0	15	−13.58	ns
TC55328-35	0	15	−13.58	ns
TMS6789-35	0	8	−20.58	ns
TC5588-35	0	10	−18.58	ns
TMS6716-35	0	10	−18.58	ns

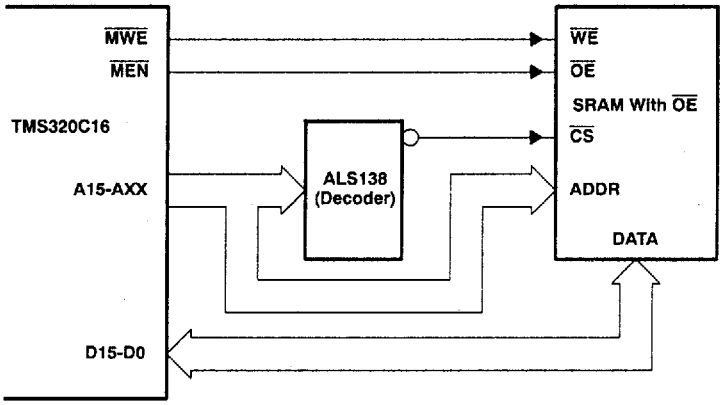


Figure 14.

static memory with chip enable and write enable

Without a separate output enable, a faster SRAM is required. Logic is added to decode address and memory control to perform a read/write cycle. The $\overline{MWE}$ signal is directly connected to the WE input of the SRAM to meet the t_v specification (see Figure 15).

Product	t_{ddeco}	t_{dmemh}	t_{dconf}	Units
CY7C164-25	7.5	10	– 11.08	ns

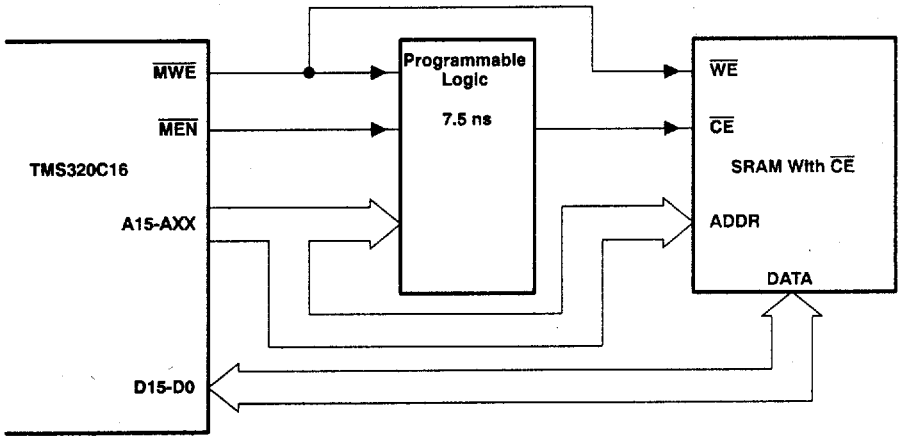


Figure 15.

EPROM Interface

The following high-speed EPROMs can be used directly:

Product	t_{ddeco}	t_{dmemh}	t_{dconf}	Units
CY7C291-35	0	25	– 3.58	ns
TMS27C291-35	0	25	– 3.58	ns

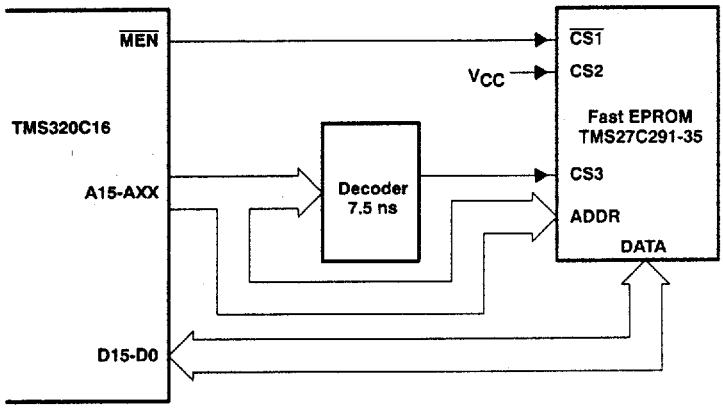


Figure 16.

interfacing latches to the TMS320C16

As with the previous devices, the memory control signal must be directly connected to the latch and the latch needs to have a separate chip select. There are several devices with this feature, including the SN74ALS996. The SN74ALS996 is an 8-bit D-type edge-triggered read-back latch with three-state outputs, connected to the 'C16 as illustrated in Figure 17.

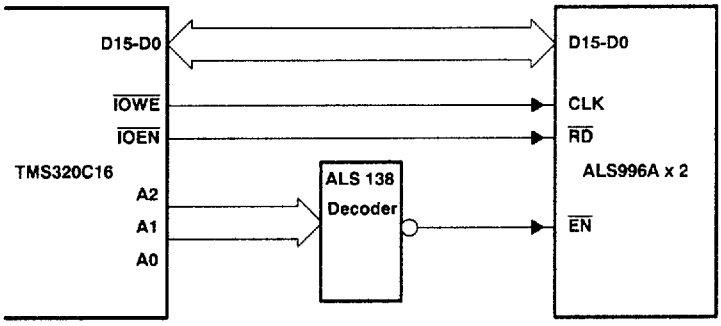
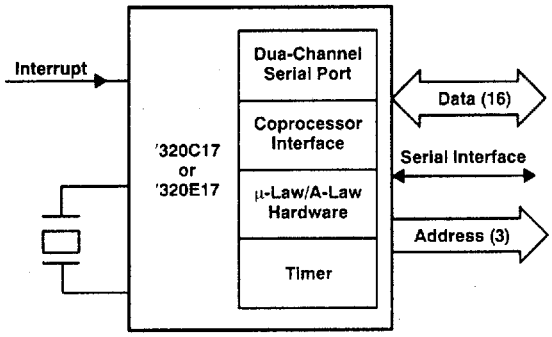


Figure 17.

JANUARY 1987 — REVISED JULY 1991

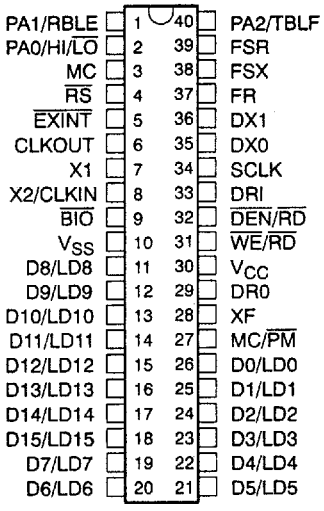
Key Features: TMS320C17/E17/LC17/P17

- 200-ns Instruction Cycle Timing (TMS320C17/E17/P17)
- 278-ns Instruction Cycle Timing (TMS320LC17)
- 256 Words of On-Chip Data RAM
- 4K Words of On-Chip Program ROM (TMS320C17/LC17)
- 4K Words of On-Chip Program EPROM (TMS320E17/P17)
- One-Time Programmable (OTP) Windowless EPROM Version Available (TMS320P17)
- EPROM Code Protection for Copyright Security
- Dual-Channel Serial Port for Full-Duplex Serial Communication
- Serial Port Timer for Standalone Serial Communication
- On-Chip Companding Hardware for μ-law/A-law PCM Conversions

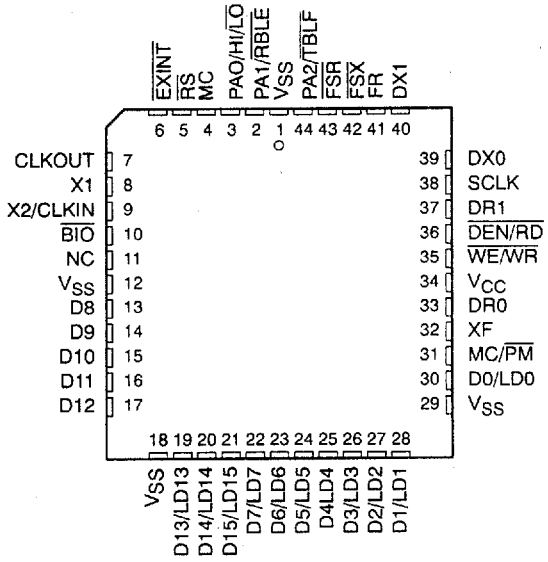


- Device Packaging:
  - 40-Pin DIP (All Devices)
  - 44-Lead PLCC (TMS320C17/LC17/P17
  - 44-Lead CER-QUAD (TMS320E17)
- 3.3 -V Low-Power Version Available (TMS320LC17)
- Operating Free-Air Temperature Range . . . 0°C to 70°C
- 16-Bit Coprocessor Interface for Common 4/8/16/32-Bit Microcomputers/Microprocessors

TMS320C17/E17/LC17/P17
N/JD Package
(Top View)



TMS320C17, TMS320E17
FN/FZ Packages
(Top View)



architecture

The 'C17/E17/LC17/P17 consists of five major functional units: the 'C15 microcomputer, a system control register, a full-duplex dual-channel serial port, companding hardware, and a coprocessor port.

Three of the I/O ports are used by the serial port, companding hardware, and the coprocessor port. Their operation is determined by the 32 bits of the system control register (see Table 6 for the control register bit definitions). Port 0 accesses control register 0 and consists of the lower 16 register bits (CR15-CR0), and is used to control the interrupts, serial port connections, and companding hardware operation. Port 1 accesses control register 1, consisting of the upper 16 control bits (CR31-CR16), as well as both serial port channels, the companding hardware, and the coprocessor port channels. Communication with the control register is via IN and OUT instructions to ports 0 and 1

Interrupts fully support the serial port interface. Four maskable interrupts ($\overline{\text{EXINT}}$, FR, $\overline{\text{FSX}}$, and $\overline{\text{FSR}}$) are mapped into I/O port 0 via control register 0. When disabled, these interrupts may be used as single-bit logic inputs polled by software.

serial port

The dual-channel serial port is capable of full-duplex serial communication and offers direct interface to two combo-codecs. Two receive and two transmit registers are mapped into I/O port 1, and operate with 8-bit data samples. Internal and external framing signals for serial port transfers (MSB first) are selected via the system control register. The serial port clock, SCLK, provides the bit timing for transfers with the serial port, and may be either an input or output. As an input, an external clock provides the timing for data transfers and framing pulse synchronization. As an output, SCLK provides the timing for standalone serial communication and is derived from the 'C17/E17/P17 system clock, X2/CLKIN, and system control register bits CR27-CR24 (see Table 7 for the available divide ratios). The internal framing (FR) pulse frequency is derived from the serial port clock (SCLK) and system control register bits CR23-CR16. This framing pulse signal provides framing pulses for combo-codecs, for a sample clock for voice-band systems, or for a timer used in control applications.

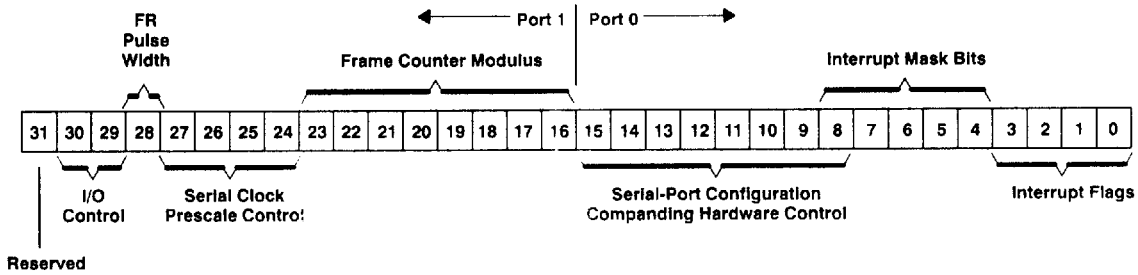
μ-law/A-law companding hardware

The 'C17/E17/LC17/P17 features hardware companding logic and can operate in either μ-law or A-law format with either sign-magnitude or twos-complement numbers. Data may be companded in either a serial mode for operation on serial port data or a parallel mode for computation inside the device. The companding logic operation is selected through CR14. No bias is required when operating in twos-complement. A bias of 33 is required for sign-magnitude in μ-law companding. Upon reset, the device is programmed to operate in sign-magnitude mode. This mode can be changed by modifying control bit 29 (CR29) in control register 1. For further information on companding, see the *TCM29C13/TCM29C14/TCM29C16/TCM29C17 Combined Single-Chip PCM Codec and Filter Data Sheet*, and the application report, *"Companding Routines for the TMS32010/TMS32020,"* in the book *Digital Signal Processing Applications with the TMS320 Family* (SPRA012A), both documents published by Texas Instruments

In the serial mode, sign-magnitude linear PCM (13 magnitude bits plus 1 sign bit for μ-law format or 12 magnitude bits plus 1 sign bit for A-law format) is compressed to 8-bit sign-magnitude logarithmic PCM by the encoder and sent to the transmit register for transmission on an active framing pulse. The decoder converts 8-bit sign-magnitude log PCM from the serial port receive registers to sign-magnitude linear PCM.

In the parallel mode, the serial port registers are disabled to allow parallel data from internal memory to be encoded or decoded for computation inside the device. In the parallel encode mode, the encoder is enabled and a 14-bit sign-magnitude value written to port 1. The encoded value is returned with an IN instruction from port 1. In the parallel decode mode, the decoder is enabled and an 8-bit sign-magnitude log PCM value is written to port 1. On the successive IN instruction from port 1, the decoded value is returned. At least one instruction should be inserted between an OUT and the successive IN when companding is performed with twos-complement values.

Table 6. Control Register Configuration



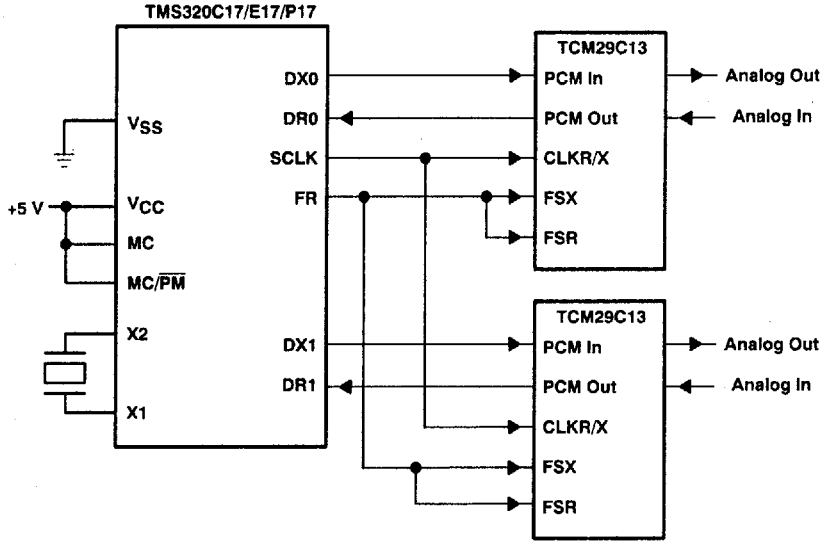
BIT	DESCRIPTION AND CONFIGURATION
0	$\overline{\text{EXINT}}$ Interrupt flag†
1	$\overline{\text{FSR}}$ interrupt flag†
2	$\overline{\text{FSX}}$ interrupt flag†
3	FR interrupt flag†
4	$\overline{\text{EXINT}}$ interrupt enable mask. When set to logic 1, an interrupt on $\overline{\text{EXINT}}$ activates device interrupt circuitry.
5	$\overline{\text{FSR}}$ interrupt enable mask. Same as $\overline{\text{EXINT}}$ control.
6	$\overline{\text{FSX}}$ interrupt enable mask. Same as $\overline{\text{EXINT}}$ control.
7	FR interrupt enable mask. Same as $\overline{\text{EXINT}}$ control.
8	Port 1 configuration control: 0 = port 1 connects to either serial-port registers or companding hardware. 1 = port 1 accesses CR31-CR16.
9	External framing enable: 0 = serial-port data transfers controlled by active FR. 1 = serial-port data transfers controlled by active $\overline{\text{FSX}}$/$\overline{\text{FSR}}$.
10	XF external logic output flag latch
11	Serial-port enable: 0 = Parallel companding mode; serial port disabled. 1 = serial companding mode; serial port registers enabled.
12	μ-law/A-law encoder enable: 0 = disabled. 1 = data written to port 1 is μ-law or A-law encoded.
13	μ-law/A-law decoder enable: 0 = disabled. 1 = data written to port 1 is μ-law or A-law decoded.
14	μ-law/A-law decoder encode/decoded select: 0 = companding hardware performs μ-law conversion. 1 = companding hardware performs A-law conversion.
15	Serial clock control: 0 = SCLK is an output, derived from the prescaler in timing logic. 1 = SCLK is an input that provides the clock for serial port and frame counter in timing logic.
23-16	Frame counter modulus. Controls FR frequency = SCLK/(CNT + 2) where CNT is binary value fo CR23-CR16‡
27-24	SCLK prescale cotnrol bits. (See Table 7 for divide ratios.)
28	FR pulse-width control: 0 = fixed-data rate; FR is 1 SCLK cycle wide. 1 = variable-data rate; FR is 8 SCLK cycles wide.
29	Two's-complement μ-law/A-law conversion enable: 0 = sign-magnitude companding 1 = twos-complement companding
30	8/16-bit length coprocessor mode select: 0 = 8-bit byte length 1 = 16-bit word length
31	Reserved for future expansion: Should be set to zero.

† Interrupt flag is cleared by writing a logic 1 to the bit with an OUT instruction to port 0.
‡ All ones in CR23-CR16 indicate a degenerative state and should be avoided. Bits are operational whether SCLK is an input or an output. CNT must be greater than 7.

Table 7. Serial Clock (SCLK) Divide Ratios (X2/CLKIN = 20.48 MHz)

CR27	CR26	CR25	CR24	DIVIDE RATIO	SCLK FREQUENCY	UNIT
0	0	0	0	32	0.640	MHz
0	0	0	1	28	0.731	MHz
0	0	1	0	24	0.853	MHz
0	1	0	0	20	1.024	MHz
1	0	0	0	16	1.280	MHz
1	0	0	1	14	1.463	MHz
1	0	1	0	12	1.706	MHz
1	1	0	0	10	2.048	MHz

The specification for μ-law and A-law log PCM coding is part of the CCITT G.711 recommendation. The following diagram shows a 'C17/E17/P17 interface to two codecs as used for μ-law or A-law companding format.



coprocessor port

The coprocessor port, accessed through I/O port 5 using IN and OUT instructions, provides a direct connection to most 4/8-bit microcomputers and 16/32-bit micorprocessors. The coprocessor interface allows the 'C17/E17/P17 to act as a peripheral (slave) microcomputer to a microprocessor, or a master to a peripheral microcomputer such as TMS7042. The coprocessor port is enabled by setting MC/$\overline{PM}$ and MC low. The microcomputer mode is enabled by setting these two pins high. (Note that MC/$\overline{PM}$ ≠ MC is undefined.)
In the microcomputer mode, the 16 data lines are used for the 6 parallel 16-bit I/O ports.

In the coprocessor mode, the 16-bit coprocessor port is reconfigured to operate as a 16-bit latched bus interface. Control bit 30 (CR30) in control register 1 is used to configure the coprocessor port to either an 8-bit or a 16-bit length. When CR30 is high, the coprocessor port is 16 bits wide thereby making all 16 bits of the data port available for 16-bit transfers to 16 and 32-bit microprocessors. When CR30 is low, the port is 8-bits wide and mapped to the low byte of the data port for interfacing to 8-bit microcomputers. When operating in the 8-bit mode, both halves of the 16-bit latch can be addressed using the HI/$\overline{LO}$ pin, thus allowing 16-bit transfers over 8 data lines. When not in the coprocessor mode, port 5 can be used as a generic I/O port.

coprocessor port (continued)

The external processor recognizes the coprocessor interface in which both processors run asynchronously as a memory-mapped I/O operation. The external processor lowers the $\overline{WR}$ line and places data on the bus. It next raises the $\overline{WR}$ line to clock the data into the on-chip latch. The rising edge of $\overline{WR}$ automatically creates an interrupt to the 'C17/E17/P17, and the falling edge of $\overline{WR}$ clears the $\overline{RBLE}$ (receive buffer latch empty) flag. When the 'C17/E17/P17 reads the coprocessor port, it causes the $\overline{RBLE}$ signal to transition to a logic low state that clears the data in the latch, and allows the interrupt condition to be cleared internally. Likewise, the external processor reads form the latch by driving the $\overline{RD}$ line active low, thus enabling the output latch to drive the latched data. When the data has been read, the external device will again bring the $\overline{RD}$ line high. This activates the $\overline{BIO}$ line to signal that the transfer is complete and the latch is available for the next transfer. The falling edge of $\overline{RD}$ resets the $\overline{TBLF}$ (transmit buffer latch full) flag. Note that the $\overline{EXINT}$ and $\overline{BIO}$ lines are reserved for coprocessor interface and cannot be driven externally when in the coprocessor mode.

An example of the use of a coprocessor interface is shown in Figure 18, in which the 'C17/E17/P17 are DSPs interfaced to the TMS70C42, an 8-bit microcontroller.

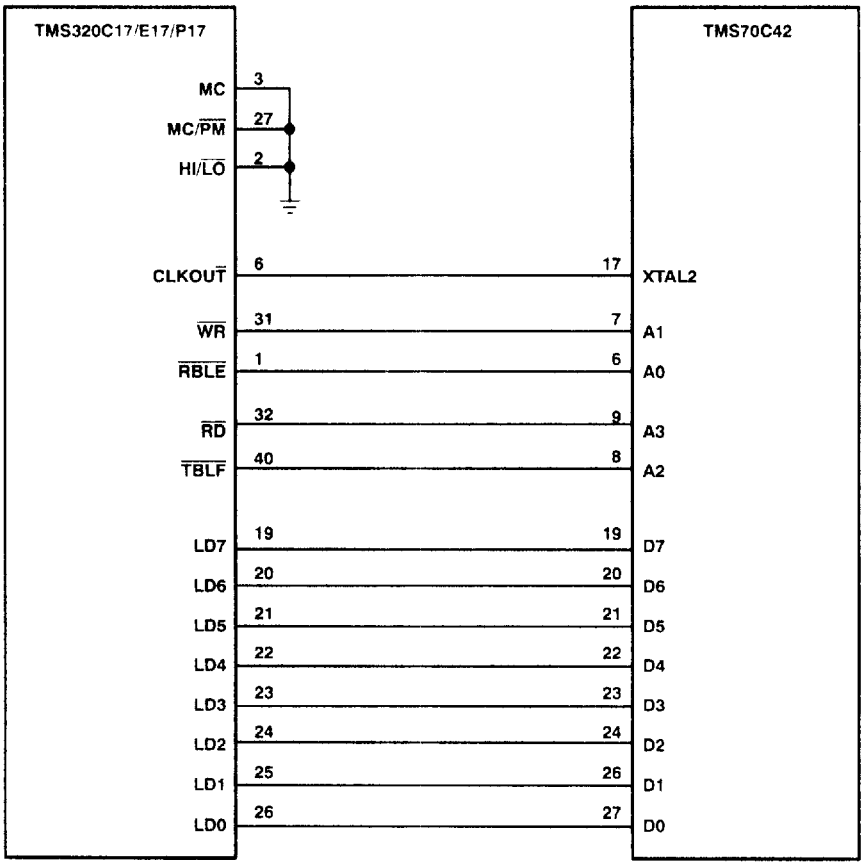


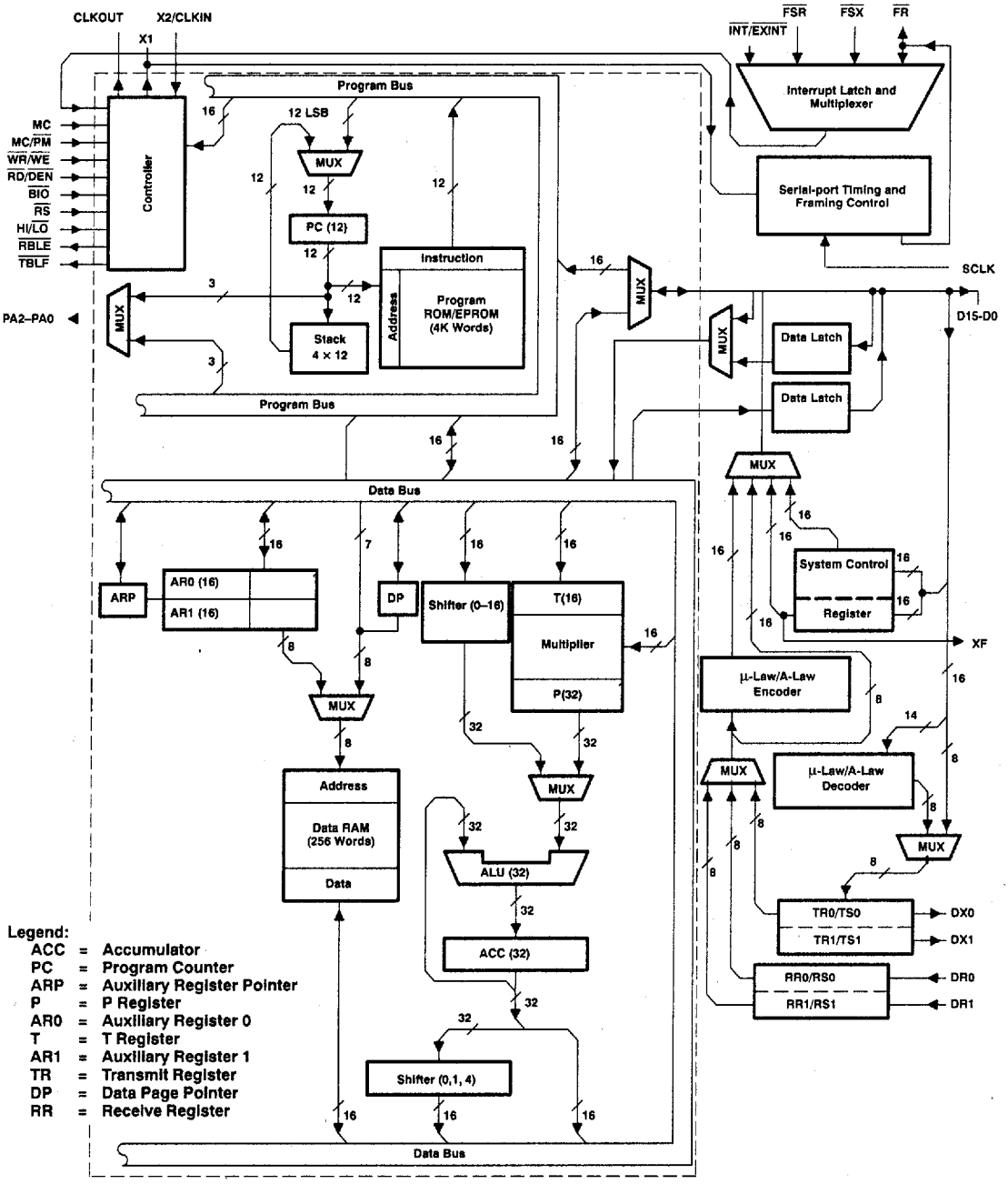
Figure 18. Coprocessor Interface

TERMINAL FUNCTIONS†

NAME	I/O‡	DEFINITION
$\overline{BIO}$	I	External polling input
CLKOUT	O	System clock output, 1/4 crystal/CLKIN frequency
D15/LD15-D0/LD0	I/O	16-bit parallel data bus/data lines for coprocessor latch
$\overline{DEN}/\overline{RD}$	I/O	Data enable for device input data/external read for output latch
DR1, DR0	I	Serial-port receive-channel inputs
DX1, DX0	O	Serial-port transmit-channel outputs
$\overline{EXINT}$	I	External interrupt input
FR	O	Internal serial-port framing output
$\overline{FSR}$	I	External serial-port receive framing input
$\overline{FSX}$	I	External serial-port transmit framing input
MC	I	Microcomputer select (must be same state as MC/$\overline{PM}$)
MC/$\overline{PM}$	I	Microcomputer/peripheral coprocessor select (must be same state as MC)
PA0/HI/$\overline{LO}$	I/O	I/O port address output/latch byte select pin
PA1/$\overline{RBLE}$	O	I/O port address output/receive buffer latch empty flag
PA2/$\overline{TBLF}$	O	I/O port address output/transmit buffer latch full flag
$\overline{RS}$	I	Reset for initializing the device
SCLK	I/O	Serial-port clock
V_{CC}	I	+ 5 V Supply
V_{SS}	I	Ground
$\overline{WE}/\overline{WR}$	O	Write enable for device output data/external write for input latch
X1	O	Crystal output for internal oscillator
X2/CLKIN	I	Crystal input for internal oscillator or external oscillator system clock input
XF	O	External-flag output pin

† See EPROM programming section.
‡ Input/Output/High-impedance state.

functional block diagram



Legend:

ACC	=	Accumulator
PC	=	Program Counter
ARP	=	Auxiliary Register Pointer
P	=	P Register
AR0	=	Auxiliary Register 0
T	=	T Register
AR1	=	Auxiliary Register 1
TR	=	Transmit Register
DP	=	Data Page Pointer
RR	=	Receive Register

electrical specifications

This section contains the electrical specifications for all versions of the 'C17/E17/P17 digital signal processors, including test parameter measurement information. Parameters with PP subscripts apply only to the 'E17/P17 in the EPROM programming mode (see Note 11).

absolute maximum ratings over operating free-air temperature range (unless otherwise noted)†

Supply voltage range, V_{CC}, except for the '320LC17 (see Note 6) . . . . . – 0.3 V to 7 V
Supply voltage range, V_{PP} . . . . . – 0.6 V to 14 V
Input voltage range . . . . . – 0.3 V to 14 V
Output voltage range . . . . . – 0.3 V to 7 V
Continuous power dissipation . . . . . 1.5 W
Operating free-air temperature: L suffix . . . . . 0°C to 70°C
A suffix . . . . . – 40°C to 85°C
Storage temperature . . . . . – 55 °C to 150 °C

† Stresses beyond those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. This is a stress rating only, and functional operation of the device at these or any other conditions beyond those indicated in the "Recommended Operating Conditions" section of this specification is not implied. Exposure to absolute-maximum-rated conditions for extended periods may affect device reliability.

NOTE 6: All voltage values are with respect to V_{SS}.

recommended operating conditions

			MIN	NOM	MAX	UNIT
V_{CC}	Supply voltage	EPROM devices	4.75	5	5.25	V
		All other devices	4.5	5	5.5	V
V_{PP}	Supply voltage (see Note 11)		12.25	12.5	12.75	V
V_{SS}	Supply voltage			0		V
V_{IH}	High-level input voltage	All inputs except CLKIN	2			V
		CLKIN	3			V
V_{IL}	Low-level input voltage	All inputs except MC/$\overline{MP}$			0.8	V
		MC/$\overline{MP}$			0.6	V
I_{OH}	High-level output current, all outputs				–300	μA
I_{OL}	Low-level output current (All outputs)				2	mA
T_A	Operating free-air temperature	L suffix	0		70	°C
		A suffix	– 40		85	°C

NOTE 11: V_{PP} can be applied only to programming pins designed to accept V_{PP} as an input. During programming the total supply current is $I_{PP} + I_{CC}$.

electrical characteristics over specified temperature range (unless otherwise noted)

PARAMETER		TEST CONDITIONS	MIN	TYP†	MAX	UNIT
I_{CC}‡ Supply current	TMS320C17	f = 20.5 MHz, V_{CC} = 5.5 V, T_A = 0°C to 70°C		50	65	mA
	TMS320E17/P17	f = 25.6 MHz, V_{CC} = 5.5 V, T_A = – 40°C to 85°C		55	75	

† All typical values are at T_A = 70°C and are used for thermal resistance calculations.
‡ I_{CC} characteristics are inversely proportional to temperature. For I_{CC} dependance on temperature, frequency, and loading, see Figure 3.

CLOCK CHARACTERISTICS AND TIMING

The 'C17/E17/P17 can use either its internal oscillator or an external frequency source for a clock.

internal clock option

The internal oscillator is enabled by connecting a crystal across X1 and X2/CLKIN (see Figure 1). The frequency of CLKOUT is one-fourth the crystal fundamental frequency. The crystal should be fundamental mode, and parallel resonant, with an effective series resistance of 30 ohms, a power dissipation of 1 mW, and should be specified at a load capacitance of 20 pF.

PARAMETER		TEST CONDITIONS	MIN	NOM	MAX	UNIT
Crystal frequency, f_x	TMS320C17	T_A = 0°C to 70°C	6.7		20.5	MHz
	TMS320E17/P17	T_A = – 40°C to 85°C	6.7		20.5	
C1, C2		T_A = 0°C to 70°C		10		pF

external clock option

An external frequency source can be used by injecting the frequency directly into X2/CLKIN with X1 left unconnected. The external frequency injected must conform to the specifications listed in the table below.

switching characteristics over recommended operating conditions

PARAMETER		TEST CONDITIONS	MIN	NOM	MAX	UNIT
$t_{c(C)}$	CLKOUT cycle time§	R_L = 825 Ω, C_L = 100 pF (see Figure 2)	195.12	200		ns
$t_{r(C)}$	CLKOUT rise time			10¶		ns
$t_{f(C)}$	CLKOUT fall time			8¶		ns
$t_{w(CL)}$	Pulse duration, CLKOUT low			92¶		ns
$t_{w(CH)}$	Pulse duration, CLKOUT high			90¶		ns
$t_{d(MCC)}$	Delay time, CLKIN↑ to CLKOUT↓		25¶		60¶	ns

§ $t_{c(C)}$ is the cycle time of CLKOUT, i.e., $4t_{c(MC)}$ (4 times CLKIN cycle time if an external oscillator is used).
¶ Values derived from characterization data and not tested.

timing requirements over recommended operating conditions

		MIN	NOM	MAX	UNIT
$t_{c(MC)}$	Master clock cycle time	48.78	50	150	ns
$t_{r(MC)}$	Rise time, master clock input		5†	10†	ns
$t_{f(MC)}$	Fall time, master clock input		5†	10†	ns
$t_{w(MCP)}$	Pulse duration, master clock	$0.45t_{c(MC)}$†		$0.6t_{c(MC)}$†	ns
$t_{w(MCL)}$	Pulse duration, master clock low		20†		ns
$t_{w(MCH)}$	Pulse duration, master clock high		20†		ns

† Values derived from characterization data and not tested.

MEMORY AND PERIPHERAL INTERFACE TIMING

switching characteristics over recommended operating conditions

PARAMETER		TEST CONDITIONS	MIN	TYP	MAX	UNIT
t_{d1}	Delay time, CLKOUT↓ to address bus valid	R_L = 825 Ω, C_L = 100 pF, (see Figure 2)	10†		50	ns
t_{d4}	Delay time, CLKOUT↓ to $\overline{DEN}$↓		$1/4t_{c(C)} - 5$†		$1/4t_{c(C)} + 15$	ns
t_{d5}	Delay time, CLKOUT↓ to $\overline{DEN}$↑		−10†		15	ns
t_{d6}	Delay time, CLKOUT↓ to $\overline{WE}$↓		$1/2t_{c(C)} - 5$†		$1/2t_{c(C)} + 15$	ns
t_{d7}	Delay time, CLKOUT↓ to $\overline{WE}$↑		−10†		15	ns
t_{d8}	Delay time, CLKOUT↓ to data bus OUT valid				$1/4t_{c(C)} + 65$	ns
t_{d9}	Time after CLKOUT↓ that data bus starts to be driven		$1/4t_{c(C)} - 5$†			ns
t_{d10}	Time after CLKOUT↓ that data bus stops bieng driven				$1/4t_{c(C)}+70$†	ns
t_v	Data bus OUT valid after CLKOUT↓		$1/4t_{c(C)} - 10$			ns
$t_{h(A\text{-}WMD)}$	Address hold time after $\overline{WE}$↑, $\overline{MEN}$↑, or $\overline{DEN}$↑ (see Note 14)		0†	2†		ns
$t_{su(A\text{-}MD)}$	Address bus setup time prior to $\overline{DEN}$↓		$1/4t_{c(C)} - 45$			ns

† Values derived from characterization data and not tested.
NOTE 14: Address bus will be valid upon $\overline{WE}$↑, $\overline{MEN}$↑, or $\overline{DEN}$↑.

timing requirements over recommended operating conditions

		TEST CONDITIONS	MIN	NOM	MAX	UNIT
$t_{su(D)}$	Setup time, data bus valid prior to CLKOUT↓	R_L = 825 Ω, C_L = 100 pF (see Figure 2)	50			ns
$t_{h(D)}$	Hold time, data bus held valid after CLKOUT↓ (see Note 16)		0			ns

NOTE 16: Data may be removed from the data bus upon $\overline{DEN}$↑ preceding CLKOUT↓.

RESET ($\overline{RS}$) TIMING

switching characteristics over recommended operating conditions

PARAMETER		TEST CONDITIONS	MIN	TYP	MAX	UNIT
t_{d11}	Delay time, $\overline{DEN}$↑, and $\overline{WE}$↑ from $\overline{RS}$	R_L 825 Ω, C_L = 100 pF, (see Figure 2)			$1/2t_{c(C)}$ +50†	ns
$t_{dis(R)}$	Data bus disable time after $\overline{RS}$				$1/4t_{c(C)}$ +50†	ns
t_{d12}	Delay time from $\overline{RS}$↓ to high-impedance SCLK				200†	ns
t_{d13}	Delay time from $\overline{RS}$↓ to high-impedance DX1, DX0				200†	ns

† Values derived form characterization data and not tested.

timing requirements over recommended operating conditions

		MIN	NOM	MAX	UNIT
$t_{su(R)}$	Reset ($\overline{RS}$) setup time prior to CLKOUT (see Note 10)	50			ns
$t_{w(R)}$	$\overline{RS}$ pulse duration	$5t_{c(C)}$			ns

NOTE 10: $\overline{RS}$ can occur anytime during a clock cycle. Time given is minimum to ensure synchronous operation.

INTERRUPT ($\overline{EXINT}$) TIMING

timing requirements over recommended operating conditions

		MIN	NOM	MAX	UNIT
$t_{f(INT)}$	Fall time, $\overline{EXINT}$			15	ns
$t_{w(INT)}$	Pulse duration, $\overline{EXINT}$	$t_{c(C)}$			ns
$t_{su(INT)}$	Setup time, $\overline{EXINT}$↓ before CLKOUT↓	50			ns

IO ($\overline{BIO}$) TIMING

timing requirements over recommended operating conditions

		MIN	NOM	MAX	UNIT
$t_{f(IO)}$	Fall time, $\overline{BIO}$			15	ns
$t_{w(IO)}$	Pulse duration, $\overline{BIO}$	$t_{c(C)}$			ns
$t_{su(IO)}$	Setup time, $\overline{BIO}$↓ before CLKOUT↓	50			ns

switching characteristics over recommended operating conditions

PARAMETER		TEST CONDITIONS	MIN	TYP	MAX	UNIT
$t_{d(XF)}$	Delay time CLOCKOUT↑ to valid XF	R_L 825 Ω, C_L = 100 pF, (see Figure 2)	5†		115	ns

† Values derived form characterization data and not tested.

SERIAL PORT TIMING

switching characteristics over recommended operating conditions

PARAMETER		MIN	NOM	MAX	UNIT
$t_{d(CH-FR)}$	Internal framing (FR) delay from SCLK rising edge			70	ns
$t_{d(DX1-XL)}$	DX bit 1 valid before SCLK falling edge	20			ns
$t_{d(DX2-XL)}$	DX bit 2 valid before SCLK falling edge	20			ns
$t_{h(DX)}$	DX hold time after SCLK falling edge	$t_{c(SCLK)}/2$			ns

timing requirements over recommended operating conditions

		MIN	NOM	MAX	UNIT
$t_{c(SCLK)}$	Serial port clock (SCLK) cycle time (see Note 17)	390		4770	ns
$t_{f(SCLK)}$	Serial port clock (SCLK) fall time			30†	ns
$t_{r(SCLK)}$	Serial port clock (SCLK) rise time			30†	ns
$t_{w(SCLKL)}$	Serial port clock (SCLK) low-pulse duration (see Note 17)	185		2500	ns
$t_{w(SCLKH)}$	Serial port clock (SCLK) high-pulse duration (see Note 17)	185		2500	ns
$t_{su(FS)}$	$\overline{FSX}/\overline{FSR}$ setup time before SCLK falling edge	100			ns
$t_{su(DR)}$	DR setup time before SCLK falling edge	20			ns
$t_{h(DR)}$	DR hold time after SCLK falling edge	20			ns

† Values derived from characterization data and not tested.

NOTES: 17. Minimum cycle time is $2t_{c(C)}$ where $t_{c(C)}$ is CLKOUT cycle time.

18. The duty cycle of the serial port clock must be within 45 to 55 percent.

COPROCESSOR INTERFACE TIMING

switching characteristics over recommended operating conditions

PARAMETER		MIN	NOM	MAX	UNIT
$t_{d(R-A)}$	$\overline{RD}$ low to $\overline{TBLF}$ high			75	ns
$t_{d(W-A)}$	$\overline{WR}$ low to $\overline{RBLE}$ high			75	ns
$t_{a(RD)}$	$\overline{RD}$ low to data valid			80	ns
$t_{h(RD)}$	Data hold time after $\overline{RD}$ high	25			ns

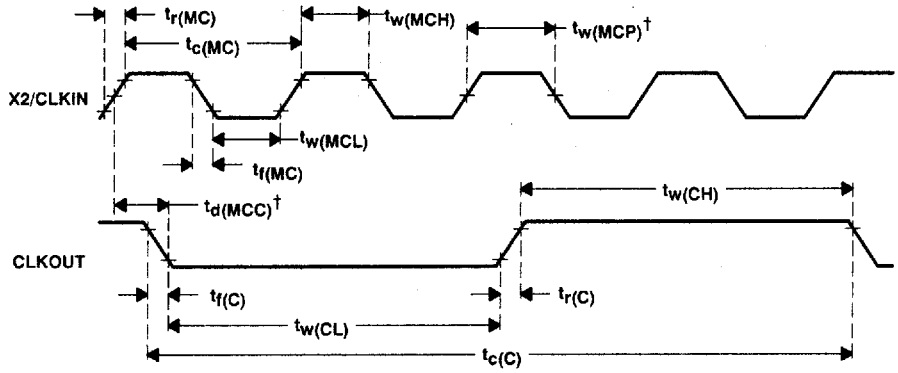
timing requirements over recommended operating conditions

		MIN	NOM	MAX	UNIT
$t_{h(HL)}$	HI/$\overline{LO}$ hold time after $\overline{WR}$ or $\overline{RD}$ high	25			ns
$t_{su(HL)}$	HI/$\overline{LO}$ setup time after $\overline{WR}$ or $\overline{RD}$ low	40			ns
$t_{su(WR)}$	Data setup time prior to $\overline{WR}$ high	30			ns
$t_{h(WR)}$	Data hold time after $\overline{WR}$ high	25			ns
$t_{w(RDL)}$	$\overline{RD}$ low-pulse duration	80			ns
$t_{w(WRL)}$	$\overline{WR}$ low-pulse duration	60			ns

TIMING DIAGRAMS

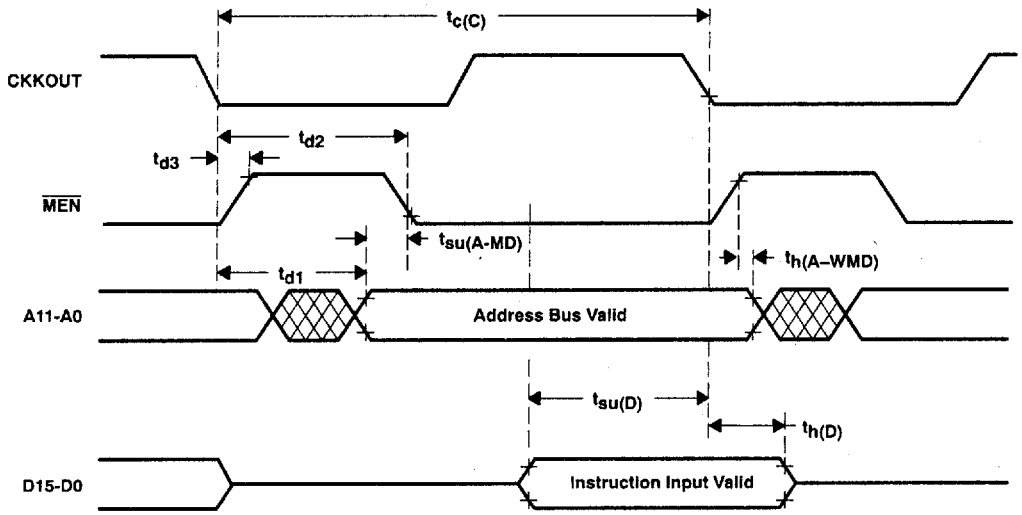
Timing measurements are referenced to and from a low voltage of 0.8 volts and a high voltage of 2 volts, unless otherwise noted.

clock timing

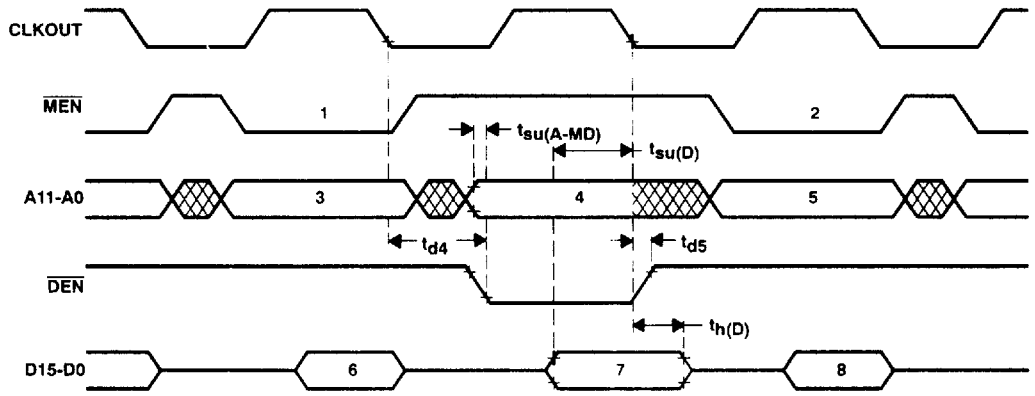


† $t_{d(MCC)}$ and $t_{w(MCP)}$ are referenced to an intermediate level of 1.5 V on the CLKIN waveform.

memory read timing



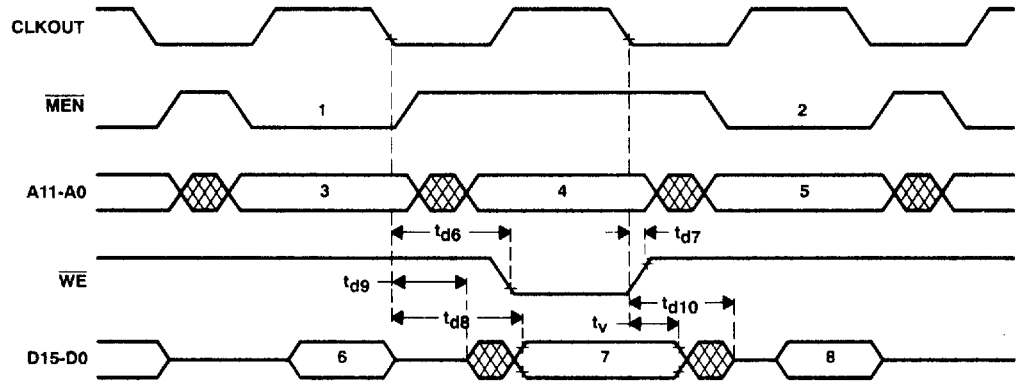
IN instruction timing



Legend:

1. IN Instruction Prefetch
2. Next Instruction Prefetch
3. Address Bus Valid
4. Peripheral Address Valid
5. Address Bus Valid
6. Instruction Input Valid
7. Data Input Valid
8. Instruction Valid

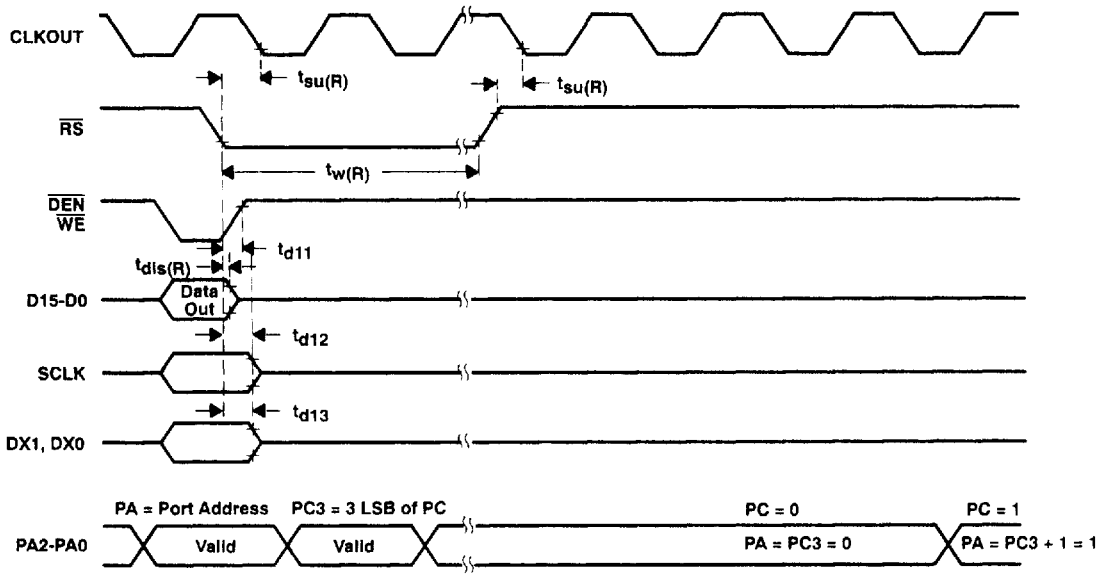
OUT instruction timing



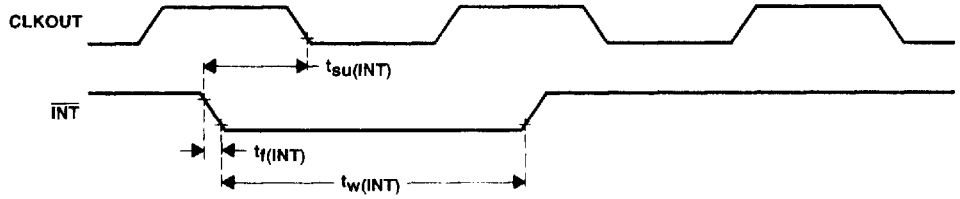
Legend:

1. OUT Instruction Prefetch
2. Next Instruction Prefetch
3. Address Bus Valid
4. Peripheral Address Valid
5. Address Bus Valid
6. Instruction Input Valid
7. Data Output Valid
8. Instruction Valid

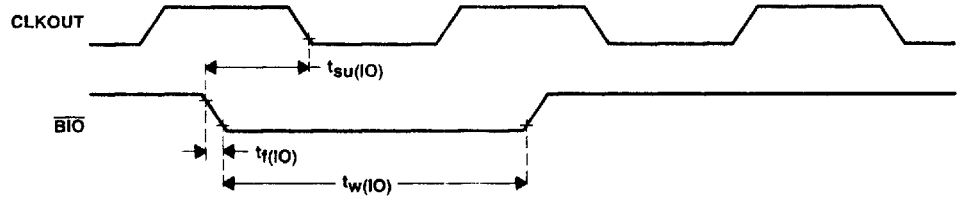
reset timing



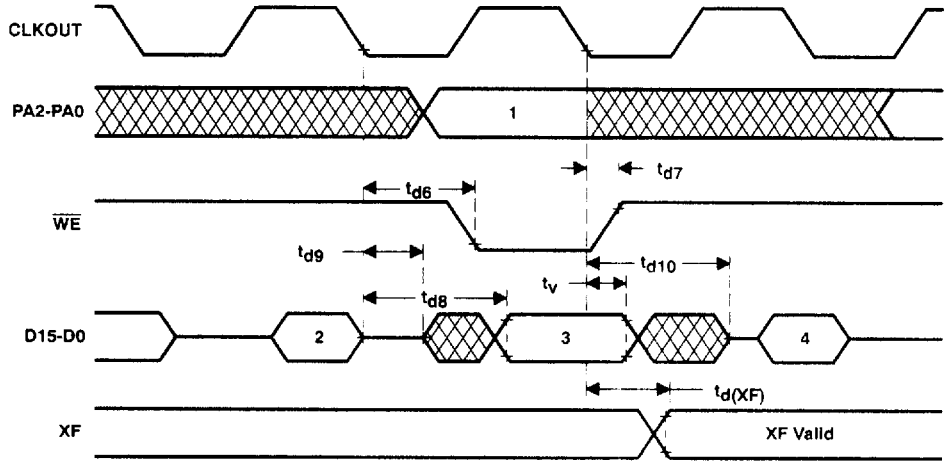
interrupt timing



$\overline{BIO}$ timing



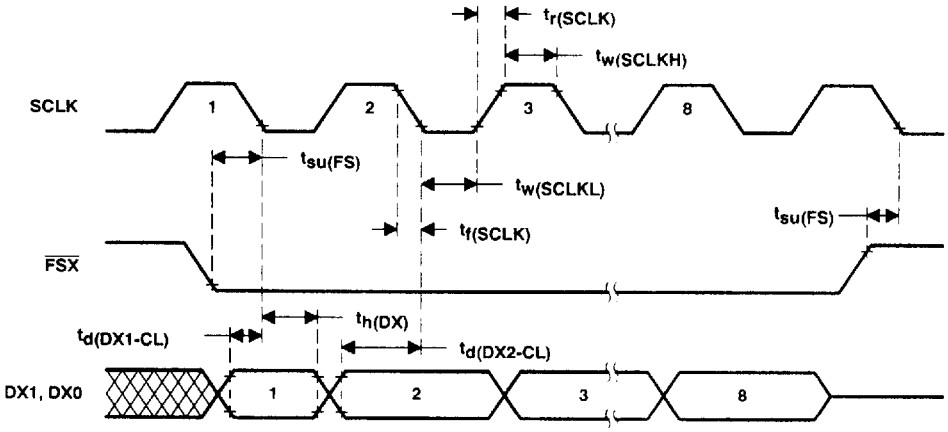
XF timing



Legend:

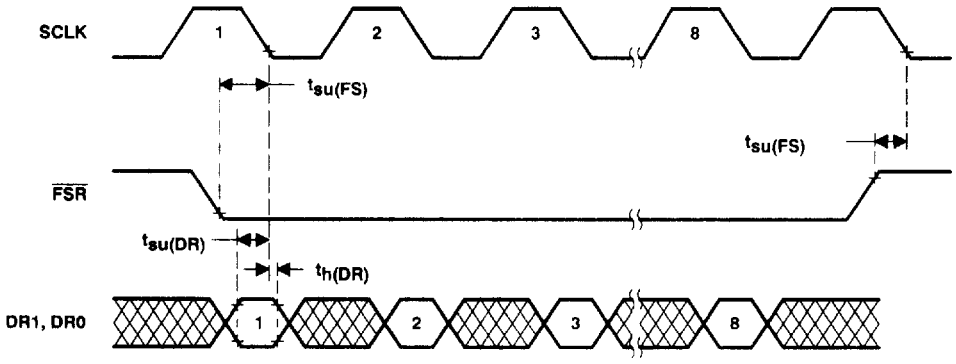
1. Port Address Valid
2. Out Opcode Valid
3. Port Data Valid
4. Next Instruction Opcode Valid

external framing: transmit timing



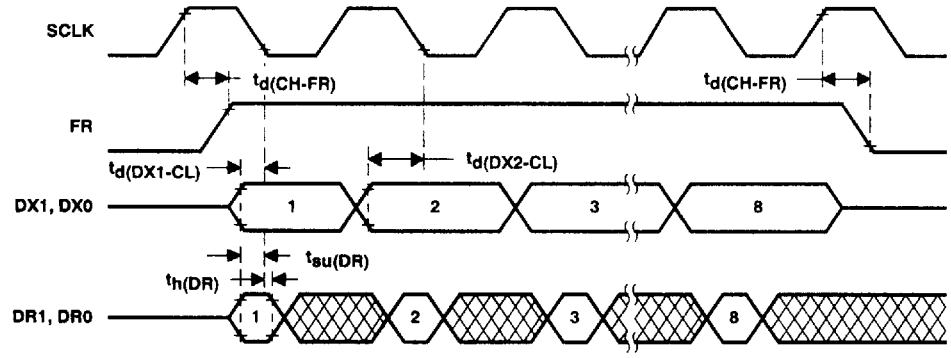
NOTES: A. Data valid on transmit output until SCLK rises.
B. The most significant bit is shifted first.

external framing: receive timing



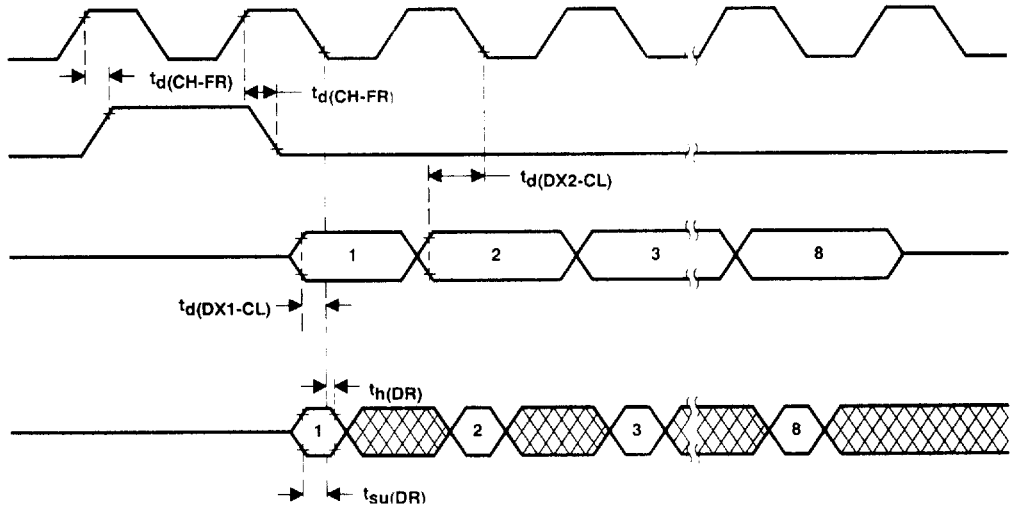
NOTE: The most significant bit is shifted first.

internal framing: variable-data rate



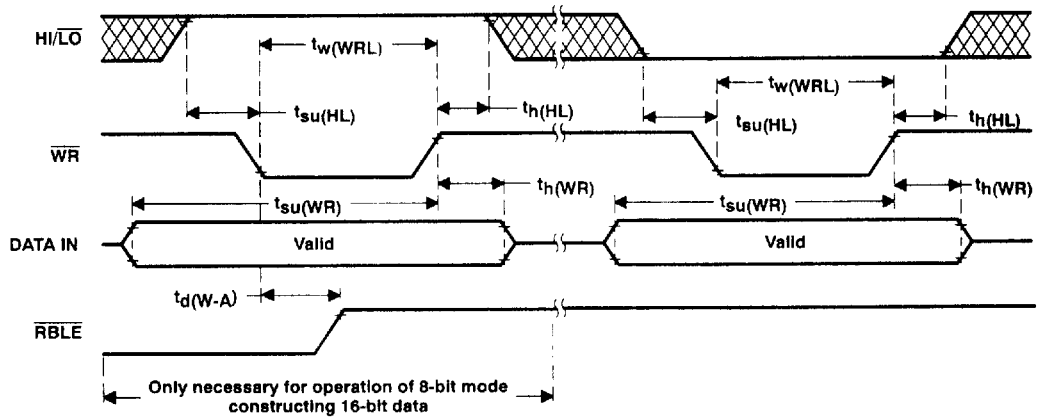
NOTE: The most significant bit is shifted first.

internal framing: fixed-data rate

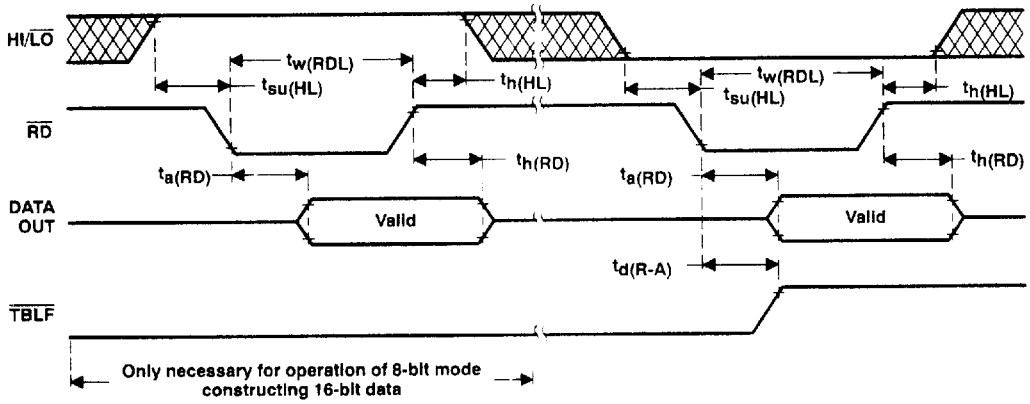


NOTE: The most significant bit is shifted first.

coprocessor timing: external write to coprocessor port



coprocessor timing: external read to coprocessor port



EPROM PROGRAMMING

absolute maximum ratings over specified temperature range (unless otherwise noted)†

Supply voltage range, V_{PP} (see Note 6) ..... −0.6 V to 14 V

† Stresses beyond those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. This is a stress rating only, and functional operation of the device at these or any other conditions beyond those indicated in the "Recommended Operating Conditions" section of this specification is not implied. Exposure to absolute-maximum-rated conditions for extended periods may affect device reliability.

NOTE 6: All voltage values are with respect to GND

recommended operating conditions

	MIN	NOM	MAX	UNIT
V_{PP} Supply voltage (see Note 11)		12.5	12.75	V

NOTE 11: V_{PP} can be applied only to programming pins designed to accept V_{PP} as an input. During programming the total supply current is $I_{PP} + I_{CC}$.

electrical characteristics over specified temperature range (unless otherwise noted)

PARAMETER	TEST CONDITIONS	MIN	TYP†	MAX	UNIT
I_{PP1} V_{PP} supply current	$V_{PP} = V_{CC} = 5.5$ V			100	µA
I_{PP2} V_{PP} supply current (during program pulse)	$V_{PP} = 12.75$ V, $V_{CC} = 5.5$ V		30	50	mA

recommended timing requirements for programming, T_A = 25°C, V_{CC} = 6 V, V_{PP} = 12.5 V, (see Note 13)

		MIN	NOM	MAX	UNIT
$t_{w(IPGM)}$	Initial program pulse duration	0.95	1	1.05	ms
$t_{w(FPGM)}$	Final pulse duration	3.8		63	ms
$t_{su(A)}$	Address setup time	2			µs
$t_{su(E)}$	$\overline{E}$ setup time	2			µs
$t_{su(G)}$	$\overline{G}$ setup time	2			µs
$t_{dis(G)}$	Output disable time from $\overline{G}$ (see Note 15)	0		130‡	ns
$t_{en(G)}$	Output enable time from $\overline{G}$			150‡	ns
$t_{su(D)}$	Data setup time	2			µs
$t_{su(VPP)}$	V_{PP} setup time	2			µs
$t_{su(VCC)}$	V_{CC} setup time	2			µs
$t_{h(A)}$	Address hold time	0			µs
$t_{h(D)}$	Data hold time	2			µs

† Values derived from characterization data and not tested.

NOTES: 13. For all switching characteristics and timing measurements, input pulse levels are 0.4 V to 2.4 V and V_{PP} = 12.5 V ± 0.25 V during programming.

15. Common test conditions apply for $t_{dis(G)}$ except during programming.

PROGRAMMING THE TMS320E17/P17 EPROM CELL

Each 'E17/P17 devices include a 4K x 16-bit industry-standard EPROM cell for prototyping, early field testing, and low-volume production. In conjunction with this EPROM, the TMS320C17 with a 4K-word masked ROM, then, provides more migration paths for cost-effective production.

Note: The TMS320P17 is a one-time programmable (OTP) EPROM device.

EPROM adapter sockets are available that provide pin-to-pin conversions for programming any 'E17/P17 devices. One adapter socket (part number RTC/PGM320C-06), shown in Figure 19, converts a 40-pin DIP into an equivalent 28-pin device. Another socket (part number RTC/PGM320C-06), not shown, permits 44- to 28-pin conversion.

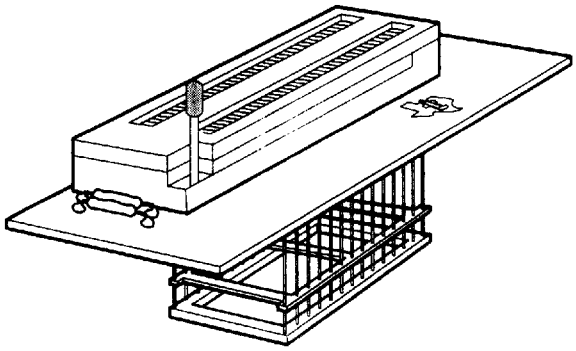
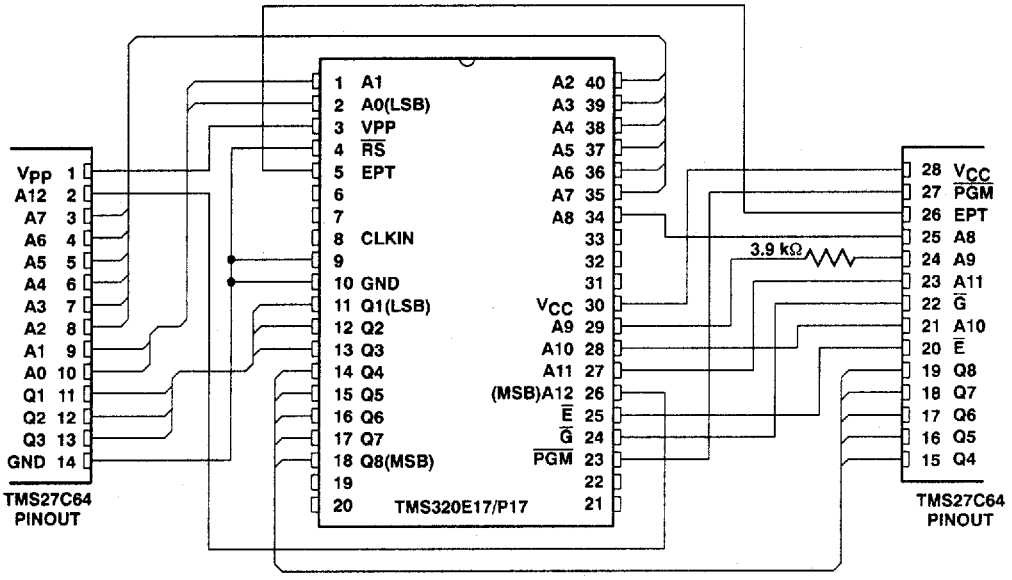


Figure 19. EPROM Adapter Socket (40-Pin to 28-Pin DIP Conversion)

Key features of the EPROM cell include the normal programming operation as well as verification. The EPROM cell also includes a code protection feature that allows code to be protected against copyright violations.

The 'E17/P17 EPROM cell is programmed using the same family and device pinout codes as the TMS27C64 8K x 8-bit EPROM. The TMS27C64 EPROM series are unltraviolet-light erasable, electrically programmable, read-only memories, fabricated using HVCMOS technology. They are pin-compatible with existing 28-pin ROMs and EPROMs. These EPROMs operate from a single 5-V supply in the read mode; however, a 12.5-V supply is needed for programming. All programming signals are TTL level. For programming outside the system, existing EPROM programmers can be used. Locations may be programmed singly, in blocks, or at random.

Figure 20 shows the wiring conversion to program the 'E17/P17 using the 28-pin pinout of the TMS27C64. Table 8 on pin nomenclature provides a description of the TMS27C64 pins. The code to be programmed into the device should be in serial mode. The 'E17/P17 devices use 13 address lines to address 4K-word memory in byte format.



CAUTION

Although acceptable by some EPROM programmers, the signature mode cannot be used on any TMS320E1x device. The signature mode will input a high-level voltage (12.5 V_{dc}) onto pin A9. Since this pin is not designed for high voltage, the cell will be damaged. To prevent an accidental application of voltage, Texas Instruments has inserted a 3.9 kΩ resistor between pin A9 of the TI programmer socket and the programmer itself.

Pin Nomenclature (TMS320E17/P17)

NAME	I/O	DEFINITION
A0-A12	I	On-chip EPROM programming address lines
CLKIN	I	Clock oscillator input
$\overline{E}$	I	EPROM chip select
EPT	I	EPROM test mode select
$\overline{G}$	I	EPROM read/verify select
GND	I	Ground
$\overline{PGM}$	I	EPROM write/program select
Q1-Q8	I/O	Data lines for byte-wide programming of on-chip 8K bytes of EPROM
$\overline{RS}$	I	Reset for initializing the device
V_{CC}	I	5-V power supply
V_{PP}	I	12.5-V power supply

Figure 20. TMS320E17/P17 EPROM Programming Conversion to TMS27C64 EPROM Pinout

Table 8 shows the programming levels required for programming, verifying, reading, and protecting the EPROM cell.

Table 8. TMS320E17/P17 Programming Mode Levels

SIGNAL NAME	TMS320E17 PIN	TMS27C64 PIN	PROGRAM	VERIFY	READ	PROTECT VERIFY	EPROM PROTECT
$\overline{E}$	25	20	V_{IL}	V_{IL}	V_{IL}	V_{IL}	V_{IH}
$\overline{G}$	24	22	V_{IH}	$\overline{PULSE}$	$\overline{PULSE}$	V_{IL}	V_{IH}
$\overline{PGM}$	23	27	$\overline{PULSE}$	V_{IH}	V_{IH}	V_{IH}	V_{IH}
V_{PP}	3	1	V_{PP}	V_{PP}	V_{CC}	V_{CC} + 1	V_{PP}
V_{CC}	30	28	V_{CC}	V_{CC}	V_{CC}	V_{CC} + 1	V_{CC} + 1
V_{SS}	10	14	V_{SS}	V_{SS}	V_{SS}	V_{SS}	V_{SS}
CLKIN	8	14	V_{SS}	V_{SS}	V_{SS}	V_{SS}	V_{SS}
$\overline{RS}$	4	14	V_{SS}	V_{SS}	V_{SS}	V_{SS}	V_{SS}
EPT	5	26	V_{SS}	V_{SS}	V_{SS}	V_{PP}	V_{PP}
Q1-Q8	11-18	11-13, 15-19	D_{IN}	Q_{OUT}	Q_{OUT}	Q8=RBIT	Q8=$\overline{PULSE}$
A0-A3	2, 1, 40, 39	10-7	ADDR	ADDR	ADDR	X	X
A4	38	6	ADDR	ADDR	ADDR	X	V_{IH}
A5	37	5	ADDR	ADDR	ADDR	X	X
A6	36	4	ADDR	ADDR	ADDR	V_{IL}	X
A7-A9	35, 34, 29	3, 25, 24	ADDR	ADDR	ADDR	X	X
A10-A12	28-26	21, 23, 2	ADDR	ADDR	ADDR	X	X

Legend:

V_{IH} = TTL high level; V_{IL} = TTL low level; ADDR = byte address bit
V_{PP} = 12.5 V ± 0.25 V; V_{CC} = 5 V ± 0.25 V; X = don't care
$\overline{PULSE}$ = low-going TTL level pulse; D_{IN} = byte to be programmed at ADDR
Q_{OUT} = byte stored at ADDR; RBIT = ROM protect bit.

programming

Since every memory bit in the cell is a logic 1, the programming operation reprograms certain bits to 0. Once programmed, these bits can be erased only by using ultraviolet light. The correct byte is placed on the data bus with V_{PP} set to the 12.5 V level. The $\overline{PGM}$ pin is then pulsed low to program in the zeroes.

erasure

Before programming, the device must be erased by exposing it to ultraviolet light. The recommended minimum exposure dose (UV-intensity × exposure-time) is 15 W•s/cm^2. A typical 12-mW/cm^2, filterless UV lamp will erase the device in 21 minutes. The lamp should be located about 2.5 cm above the chip during erasure. After exposure, all bits are in the high state.

verify/read

To verify correct programming, the EPROM cell can be read using either the verify or read line definitions shown in Table 8 assuming the inhibit bit has not been programmed.

program inhibit

Programming may be inhibited by maintaining a high level input on the $\overline{E}$ pin or $\overline{PGM}$ pin.

read

The EPROM contents may be read independent of the programming cycle, provided the RBIT (ROM protect bit) has not been programmed. The read is accomplished by setting $\overline{E}$ to zero and pulsing $\overline{G}$ low. The contents of the EPROM location selected by the value on the address inputs appear on Q8-Q1.

output disable

During the EPROM programming process, the EPROM data outputs may be disabled, if desired, by establishing the output disable state. This state is selected by setting $\overline{G}$ and $\overline{E}$ pins high. While output disable is selected, Q8-Q1are placed in the high-impedance state.

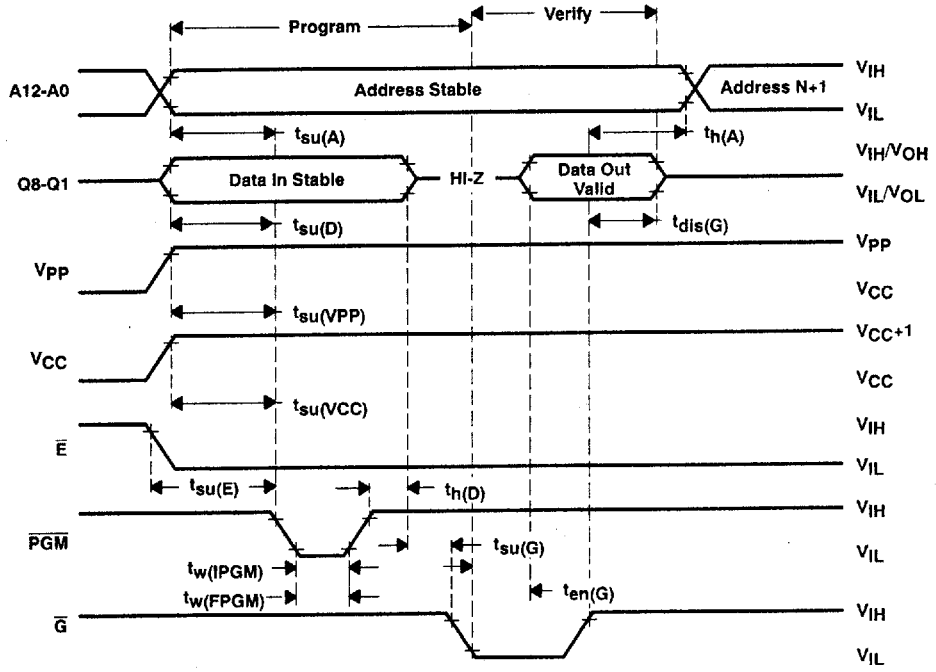
EPROM protection

To protect the proprietary algorithms existing in the code programmed on-chip, the ability to read or verify code from external accesses can be completely disabled. Programming the RBIT disables external access of the EPROM cell, making it impossible to access the code resident in the EPROM cell. The only way to remove this protection is to erase the entire EPROM cell, thus removing the proprietary information. The signal requirements for programming this bit are shown in Table 8. The cell can be determined as protected by verifying the programming of the RBIT shown in the table.

standard programming procedure

Before programming, the device must first be completely erased. The device can then be programmed with the correct code. It is advisable to program unused sections with zeroes as a further security measure. After the programming is complete, the code programmed into the cell should be verified. If the cell passes verification, the next step is to program the ROM protect bit (RBIT). Once the RBIT programming is verified, an opaque label should be placed over the window to protect the EPROM cell from inadvertent erasure by ambient light. At this point, the programming is complete, and the device is ready to be placed into its destination circuit.

program cycle timing



absolute maximum ratings over specified temperature range (unless otherwise noted)†

Supply voltage range, V_{CC} (see Note 6) ..... – 0.3 V to 4.6 V
Input voltage range ..... – 0.3 V to V_{CC} to 0.5 V
Output voltage range ..... – 0.3 V to V_{CC} to 0.5 V
Continuous power dissipation ..... 75 mW
Air temperature range above operating devices: L version ..... 0°C to 70°C
A version ..... – 40°C to 85°C
Storage temperature range ..... – 55°C to +150°C

† Stresses beyond those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. This is a stress rating only, and functional operation of the device at these or any other conditions beyond those indicated in the "Recommended Operating Conditions" section of this specification is not implied. Exposure to absolute-maximum-rated conditions for extended periods may affect device reliability.

NOTE 6: All voltage values are with respect to V_{SS}.

recommended operating conditions

			MIN	NOM	MAX	UNIT
V_{CC}	Supply voltage		3.0	3.3	3.6	V
V_{SS}	Supply voltage			0		V
V_{IH}	High-level input voltage	All inputs except CLKIN	2.0			V
		CLKIN	2.5			V
V_{IL}	Low-level input voltage	All inputs			0.55	V
I_{OH}	High-level output current (all outputs)				– 300	μA
I_{OL}	Low-level output current (all outputs)				1.5	mA
T_A	Operating free-air temperature	L version	0		70	°C
		A version	– 40		85	°C

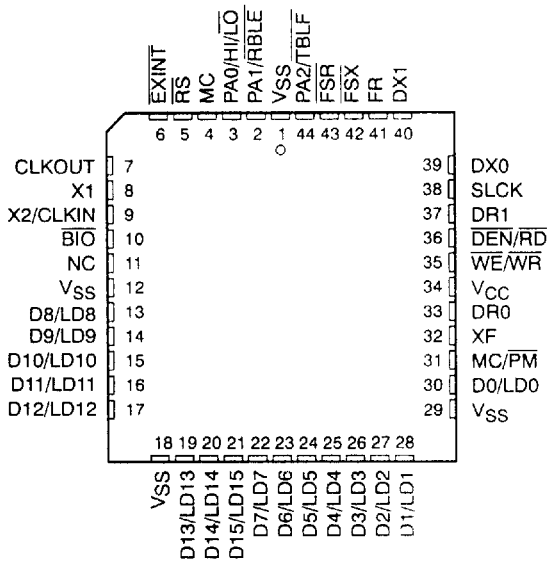
electrical characteristics over specified temperature range (unless otherwise noted)

PARAMETER			TEST CONDITIONS	MIN	TYP§	MAX	UNIT
V_{OH}	High-level output voltage		I_{OH} = MAX	2.0			V
			I_{OH} = 20 μA (see Note 19)	V_{CC} – 0.4¶			V
V_{OL}	Low-level output voltage		I_{OL} = MAX			0.5	V
I_{OZ}	Off-state ouput current		V_{CC} = MAX, V_O = V_{CC}			20	μA
			V_O = V_{SS}			–20	
I_I	Input current		V_I = V_{SS} to V_{CC}, All inputs except CLKIN			±20	μA
			V_I = V_{SS} to V_{CC}, CLKIN			±50	
C_i	Input capacitance	Data bus	f = 1 MHz, All other pins 0 V		25¶		pF
		All others			15¶		
C_o	Output capacitance	Data bus			25¶		pF
		All others			10¶		

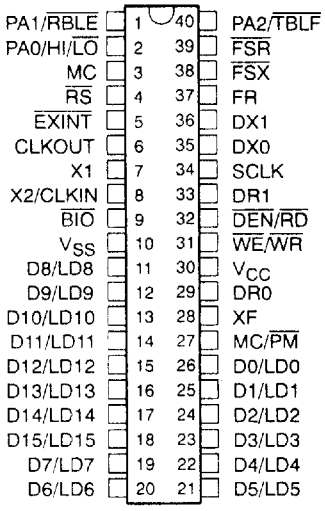
§ All typical values are at V_{CC} = 3.3 V, T_A = 25°C.
¶ Values derived from characterization data and not tested.

NOTE 19: This voltage specification is included for interface to HC logic. All other timing parameters defined in this data sheet are specified for the test load circuit shown in Figure 2.

TMS320LC17
FN PACKAGE
(TOP VIEW)



TMS320LC17
N PACKAGE
(TOP VIEW)



electrical characteristics over specified ranges (unless otherwise noted)

PARAMETER		TEST CONDITIONS	MIN	TYP†	MAX	UNIT
I_{CC}‡	Supply current	f = 14.4 MHz, V_{CC} = 3.6 V, T_A = 0°C to 70°C		15	20	mA

† All typical values are at T_A = 70°C and are used for thermal resistance calculations.
‡ I_{CC} characteristics are inversely proportional to temperature. For I_{CC} dependence on frequency, see Figure 3.

clock characteristics and timing

The TMS320LC17 can use either its internal oscillator or an external frequency source for a clock.

internal clock option

The internal oscillator is enabled by connecting a crystal across X1 and X2/CLKIN (see Figure 1). The frequency of CLKOUT is one-fourth the crystal fundamental frequency. The crystal should be fundamental mode, and parallel resonant, with an effective series resistance of 30 ohms, a power dissipation of 1 mW, and be specified at a load capacitance of 20 pF.

PARAMETER	TEST CONDITIONS	MIN	NOM	MAX	UNIT
Crystal frequency f_X	T_A = – 40°C to 85°C	4.0		14.4	MHz
C1, C2			10		pF

external clock option

An external frequency source can be used by injecting the frequency directly into X2/CLKIN with X1 left unconnected. The external frequency injected must conform to the specifications listed in the table below.

switching characteristics over recommended operating conditions

PARAMETER		TEST CONDITIONS	MIN	NOM	MAX	UNIT
$t_{c(C)}$	CLKOUT cycle time§	R_L = 825 Ω, C_L = 100 pF, (see Figure 2)	277.78		1000	ns
$t_{r(C)}$	CLKOUT rise time			10¶		ns
$t_{f(C)}$	CLKOUT fall time			8¶		ns
$t_{w(CL)}$	Pulse duration, CLKOUT low			131		ns
$t_{w(CH)}$	Pulse duration, CLKOUT high			129		ns
$t_{d(MCC)}$	Delay time CLKIN↑ to CLKOUT↓		25		75	ns

§ $t_{c(C)}$ is the cycle time of CLKOUT, i.e., $4t_{c(MC)}$ (4 times CLKIN cycle time if an external oscillator is used).
¶ Values derived from characterization data and not tested

timing requirements over recommended operating conditions

		MIN	NOM	MAX	UNIT
$t_{c(MC)}$	Master clock cycle time	69.5		150	ns
$t_{r(MC)}$	Rise time, master clock input		5†	10†	ns
$t_{f(MC)}$	Fall time, master clock input		5†	10†	ns
$t_{w(MCP)}$	Pulse duration, master clock	0.4 $t_{c(MC)}$†		0.6 $t_{c(MC)}$†	ns
$t_{w(MCL)}$	Pulse duration, master clock low at $t_{c(MC)}$ min		30		ns
$t_{w(MCH)}$	Pulse duration, master clock high at $t_{c(MC)}$ min		30		ns

† Values derived from characterization data and not tested.

MEMORY AND PERIPHERAL INTERFACE TIMING

switching characteristics over recommended operating conditions

PARAMETER		TEST CONDITIONS	MIN	MAX	UNIT
t_{d1}	Delay time CLKOUT↓ to address bus valid	R_L = 825 Ω, C_L = 100 pF, (see Figure 2)	10†	100	ns
t_{d4}	Delay time CLKOUT↓ to $\overline{DEN}$↓		1/4 $t_{c(C)}$−5†	1/4 $t_{c(C)}$+25	ns
t_{d5}	Delay time CLKOUT↓ to $\overline{DEN}$↑		−10†	30	ns
t_{d6}	Delay time CLKOUT↓ to $\overline{WE}$↓		1/2 $t_{c(C)}$−5†	1/2 $t_{c(C)}$+25	ns
t_{d7}	Delay time CLKOUT↓ to $\overline{WE}$↑		−10†	30	ns
t_{d8}	Delay time CLKOUT↓ to data bus OUT valid			1/4 $t_{c(C)}$+130	ns
t_{d9}	Time after CLKOUT↓ that data bus starts to be driven		1/4 $t_{c(C)}$−5†		ns
t_{d10}	Time after CLKOUT↓ that data bus stops being driven			1/4 $t_{c(C)}$+90	ns
t_v	Data bus OUT valid after CLKOUT↓		1/4 $t_{c(C)}$−10		ns
$t_{h(A\text{-}WMD)}$	Address hold time after WE↑, $\overline{MEN}$↑, or $\overline{DEN}$↑ (see Note 14)		0†		ns
$t_{su(A\text{-}MD)}$	Address bus setup time or $\overline{DEN}$↓		0		ns

† Values derived from characterization data and not tested.
NOTE 14: Address bus will be valid upon $\overline{WE}$↑, $\overline{MEN}$↑, or $\overline{DEN}$↑

timing requirements over recommended operating conditions

		TEST CONDITIONS	MIN	NOM	MAX	UNIT
$t_{su(D)}$	Setup time data bus valid prior to CLKOUT↓	R_L = 825 Ω, C_L = 100 pF, (see Figure 2)	80			ns
$t_{h(D)}$	Hold time data bus held valid after CLKOUT↓ (see Note 9)		0			ns

NOTE 9: Data may be removed from the data bus upon $\overline{MEN}$↑ or $\overline{DEN}$↑ preceding CLKOUT↓.

RESET ($\overline{RS}$) TIMING

switching characteristics over recommended operating conditions

PARAMETER		TEST CONDITIONS	MIN	NOM	MAX	UNIT
t_{d11}	Delay time $\overline{DEN}$↑, $\overline{WE}$↑, and $\overline{MEN}$↑ from $\overline{RS}$	R_L = 825 Ω, C_L = 100 pF, (see Figure 2)			$1/2t_{c(C)}+75$	ns
$t_{dis(R)}$	Data bus disable time after $\overline{RS}$				$1/4t_{c(C)}+75$	ns
t_{d12}	Delay time from $\overline{RS}$↓ to high-impedance SCLK				200†	ns
t_{d13}	Delay time from $\overline{RS}$↓ to high-impedance DX1, DX0				200†	ns

† These values were derived from characterization data and not tested.

timing requirements over recommended operating conditions

		MIN	NOM	MAX	UNIT
$t_{su(R)}$	Reset ($\overline{RS}$) setup time prior to CLKOUT (see Note 10)	85			ns
$t_{w(R)}$	$\overline{RS}$ pulse duration	$5t_{c(C)}$			ns

NOTE 10: $\overline{RS}$ can occur anytime during a clock cycle. Time given is minimum to ensure synchronous operation.

INTERRUPT ($\overline{EXINT}$) TIMING

timing requirements over recommended operating conditions

		MIN	NOM	MAX	UNIT
$t_{f(INT)}$	Fall time $\overline{EXINT}$			15	ns
$t_{w(INT)}$	Pulse duration $\overline{EXINT}$	$t_{c(C)}$			ns
$t_{su(INT)}$	Setup time $\overline{EXINT}$↓ before CLKOUT↓	85			ns

I/O ($\overline{BIO}$) TIMING

timing requirements over recommended operating conditions

		MIN	NOM	MAX	UNIT
$t_{f(IO)}$	Fall time $\overline{BIO}$			15	ns
$t_{w(IO)}$	Pulse duration $\overline{BIO}$	$t_{c(C)}$			ns
$t_{su(IO)}$	Setup time $\overline{BIO}$↓ before CLKOUT↓	85			ns

I/O ($\overline{BIO}$) TIMING

switching characteristics over recommended operating conditions

PARAMETER		TEST CONDITIONS	MIN	NOM	MAX	UNIT
$t_{d(XF)}$	Delay time CLKOUT↓ to valid XF	R_L = 825 Ω, C_L = 100 pF, (see Figure 2)	5†		115	ns

† Values derived from characterization data and not tested.

SERIAL PORT TIMING

switching characteristics over recommended operating conditions

		MIN	NOM	MAX	UNIT
$t_{d(CH-FR)}$	Internal framing (FR) delay from SCLK rising edge			120	ns
$t_{d(DX1-CL)}$	DX bit 1 valid before SCLK falling edge	20			ns
$t_{d(DX2-CL)}$	DX bit 2 valid before SCLK falling edge	20			ns
$t_{h(DX)}$	DX hold time after SCLK falling edge	$t_{c(SCLK)}/2$			ns

timing requirements over recommended operating conditions

		MIN	NOM	MAX	UNIT
$t_{c(SCLK)}$	Serial port clock (SCLK) cycle time‡	555		8000	ns
$t_{f(SCLK)}$	Serial port clock (SCLK) fall time			30†	ns
$t_{r(SCLK)}$	Serial port clock (SCLK) rise time			30†	ns
$t_{w(SCLK)}$	Serial port clock (SCLK) low, pulse duration§	250		4400	ns
$t_{w(SCLKH)}$	Serial port clock (SCLK) high, pulse duration§	250		4400	ns
$t_{su(FS)}$	$\overline{FSX}/\overline{FSR}$ setup time before SCLK falling edge	130			ns
$t_{su(DR)}$	DR setup time before SCLK falling edge	20			ns
$t_{h(DR)}$	DR hold time after SCLK falling edge	20			ns

† Values derived from characterization data and not tested.
‡ Minimum cycle time is $2t_{c(C)}$ where $t_{c(C)}$ is CLKOUT cycle time.
§ The duty cycle of the serial port clock must be within 45 to 55%

COPROCESSOR INTERFACE TIMING

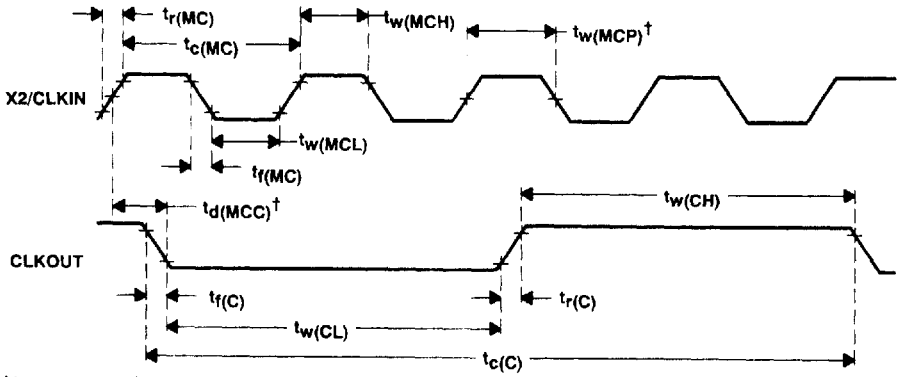
switching characteristics over recommended operating conditions

		MIN	NOM	MAX	UNIT
$t_{d(R-A)}$	$\overline{RD}$ low to $\overline{TBLF}$ high			150	ns
$t_{d(W-A)}$	$\overline{WR}$ low to $\overline{RBLF}$ high			150	ns
$t_{a(RD)}$	$\overline{RD}$ low to data valid			150	ns
$t_{h(RD)}$	Data hold time after $\overline{RD}$ high	25			

timing requirements over recommended operating conditions

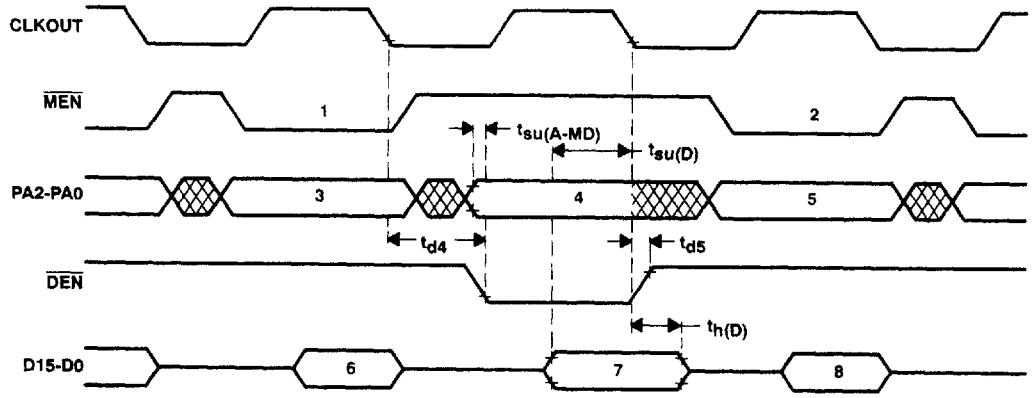
		MIN	NOM	MAX	UNIT
$t_{h(HL)}$	HI/$\overline{RD}$ hold time after $\overline{WR}$ or $\overline{RD}$ high	25			ns
$t_{su(HL)}$	HI/$\overline{RD}$ setup time prior to $\overline{WR}$ or $\overline{RD}$ low	40			ns
$t_{su(WR)}$	Data setup time prior to $\overline{WR}$ high	50			ns
$t_{h(WR)}$	Data hold time after $\overline{WR}$ high	35			ns
$t_{w(RDL)}$	Pulse duration, $\overline{RD}$ low	150			ns
$t_{w(WRL)}$	Pulse duration, $\overline{WR}$ low	150			ns

clock timing



† $t_{d(MCC)}$ and $t_{w(MCP)}$ are referenced to an intermediate level of 1.5 V on the CLKIN waveform.

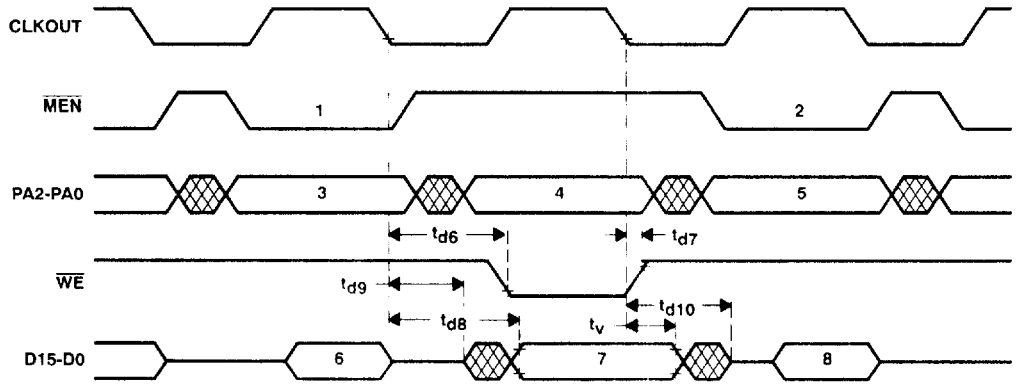
IN instruction timing



Legend:

1. IN Instruction Prefetch
2. Next Instruction Prefetch
3. Address Bus Valid
4. Peripheral Address Valid
5. Address Bus Valid
6. Instruction Valid
7. Data Input Valid
8. Instruction Valid

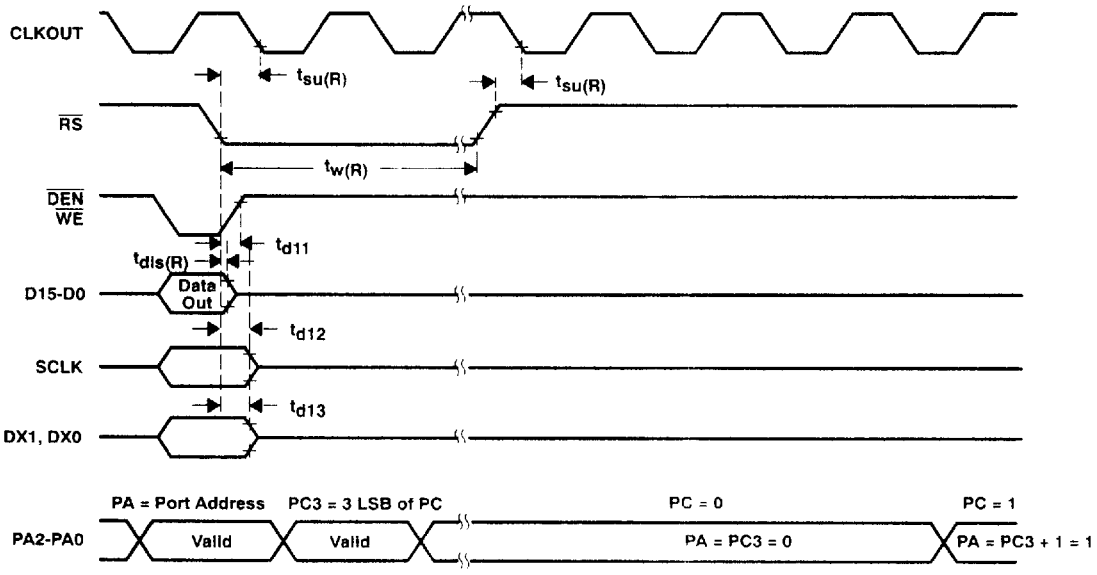
OUT instruction timing



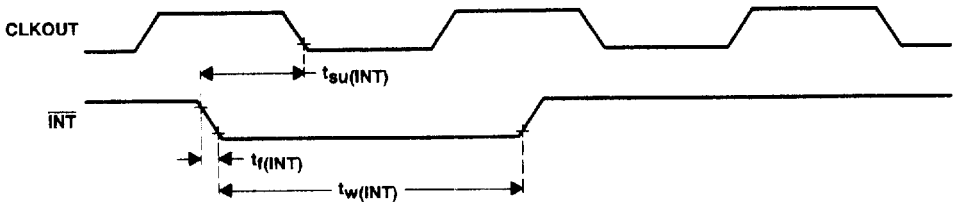
Legend:

1. OUT Instruction Prefetch
2. Next Instruction Prefetch
3. Address Bus Valid
4. Peripheral Address Valid
5. Address Bus Valid
6. Instruction Valid
7. Data Output Valid
8. Instruction Valid

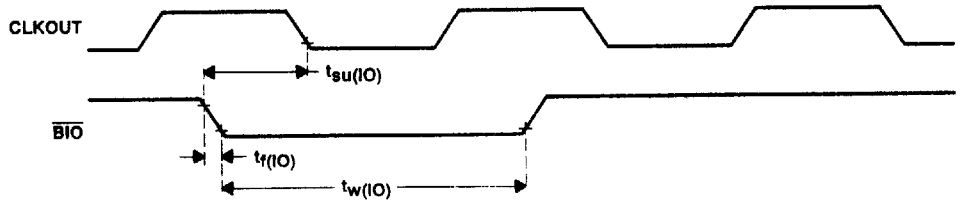
reset timing



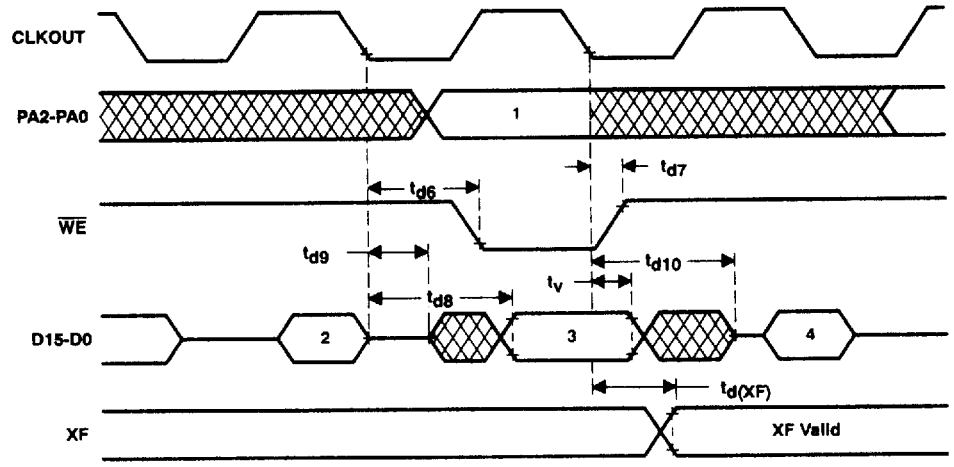
Interrupt timing



$\overline{BIO}$ timing



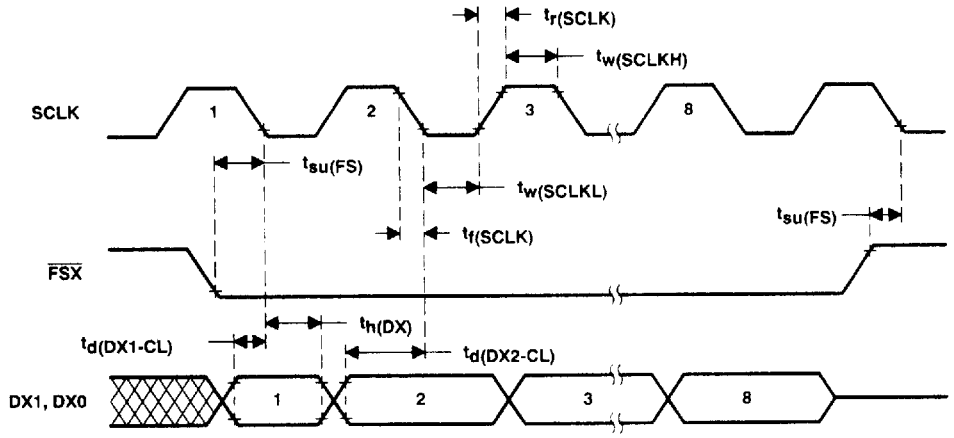
XF timing



Legend:

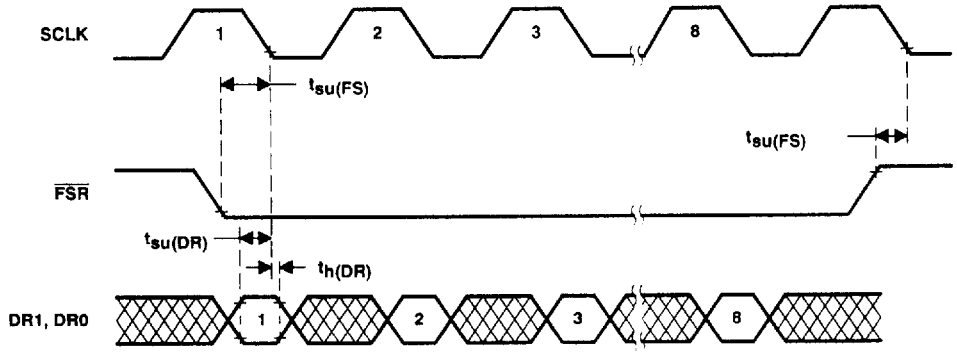
1. Port Address Valid
2. Out Opcode Valid
3. Port Data Valid
4. Next Instruction Opcode Valid

external framing: transmit timing



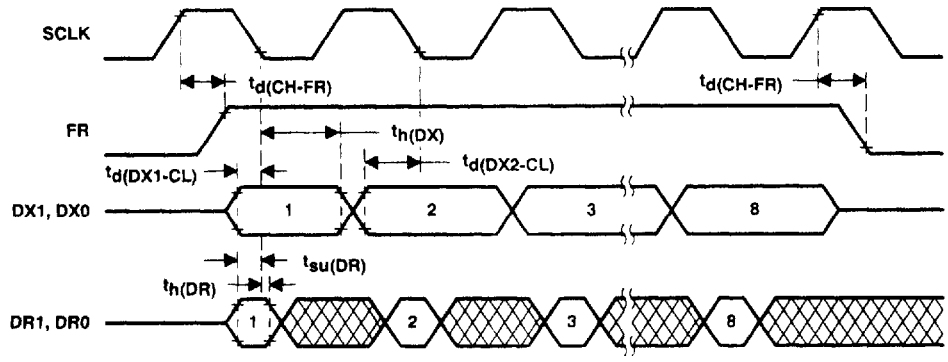
NOTES: A. Data valid on transmit output until SCLK rises.
B. The most significant bit is shifted first

external framing: receive timing



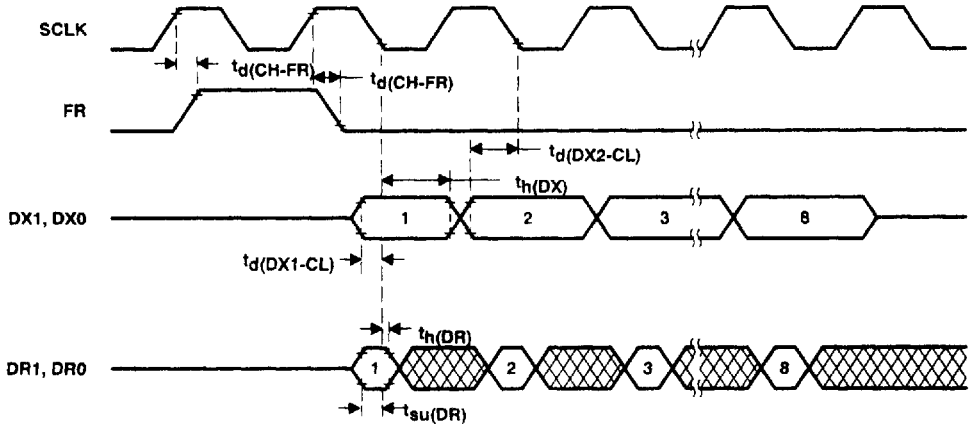
NOTE B: The most significant bit is shifted first

internal framing: variable-data rate



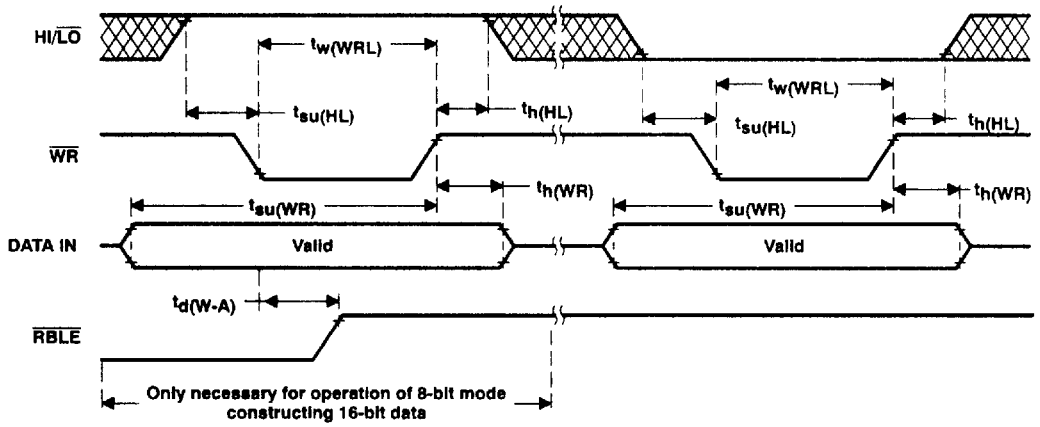
NOTE: The most significant bit is shifted first.

internal framing: fixed-data rate

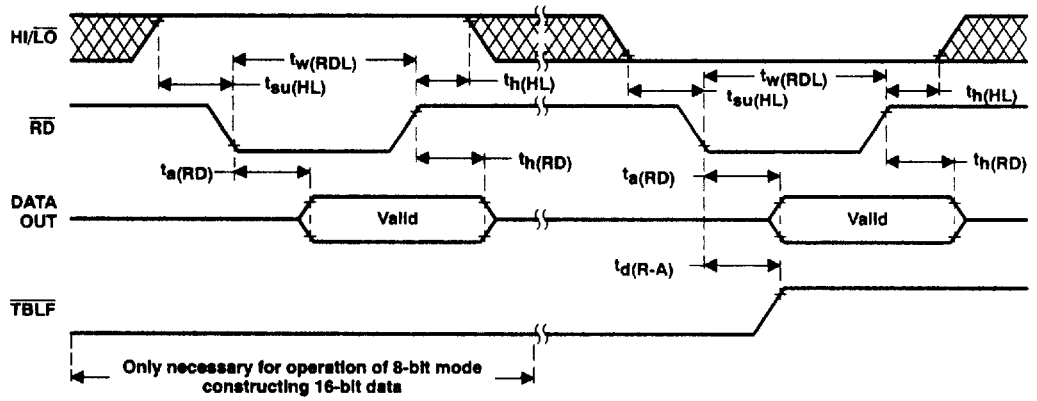


NOTE: The most significant bit is shifted first.

coprocessor timing: external write to coprocessor port



coprocessor timing: external read to coprocessor port



THERMAL RESISTANCE CHARACTERISTICS

Commercial Devices
Device/Package Thermal Resistance
Junction To Case

DEVICE	$R_{\theta JC}$ (°C/W)				
	PDIP (N)	CDIP (JD)	PLCC (FN)	CLCC (FZ)	QFP (PG)
TMS320C10	26		17		
TMS320C10-14	26		17		
TMS320C10-25	26		17		
TMS320C14			11		
TMS320E14				8	
TMS320P14			11		
TMS320C15	26		17		
TMS320C15-25	26		17		
TMS320E15		8		8	
TMS320E15-25		8		8	
TMS320LC15	26		17		
TMS320P15	13		13		
TMS320C16					25
TMS320C17	26		17		
TMS320E17		8		8	
TMS320LC17	26		17		
TMS320P17	13		13		

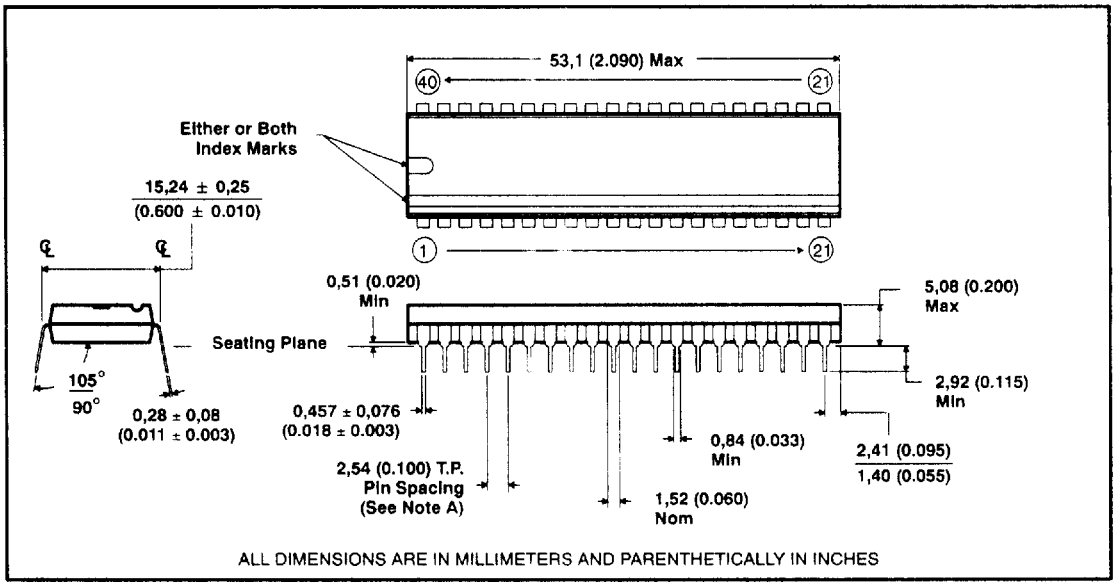
THERMAL RESISTANCE CHARACTERISTICS

Commercial Devices
Device/Package Thermal Resistance
Junction To Ambient

DEVICE	$R_{\theta JA}$ (°C/W)				
	PDIP (N)	CDIP (JD)	PLCC (FN)	CLCC (FZ)	QFP (PG)
TMS320C10	84		60		
TMS320C10-14	84		60		
TMS320C10-25	84		60		
TMS320C14			46		
TMS320E14				49	
TMS320P14			46		
TMS320C15	84		60		
TMS320C15-25	84		60		
TMS320E15		40		64	
TMS320E15-25		40		64	
TMS320LC15	84		60		
TMS320P15	40		55		
TMS320C16					120
TMS320C17	84		60		
TMS320E17		40		64	
TMS320LC17	84		60		
TMS320P17	40		55		

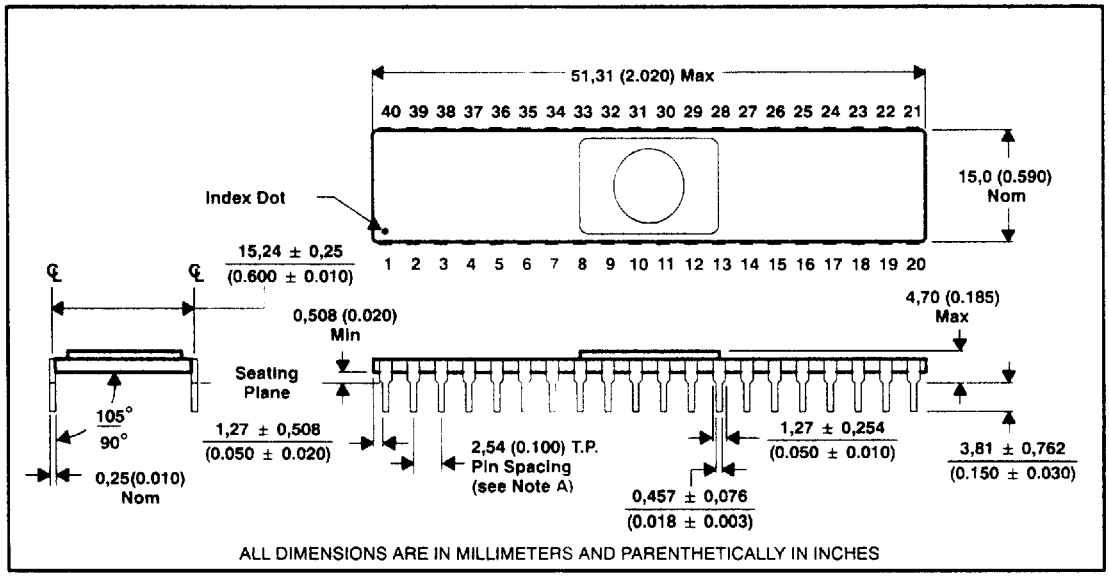
MECHANICAL DATA

40-pin plastic dual-in-line package



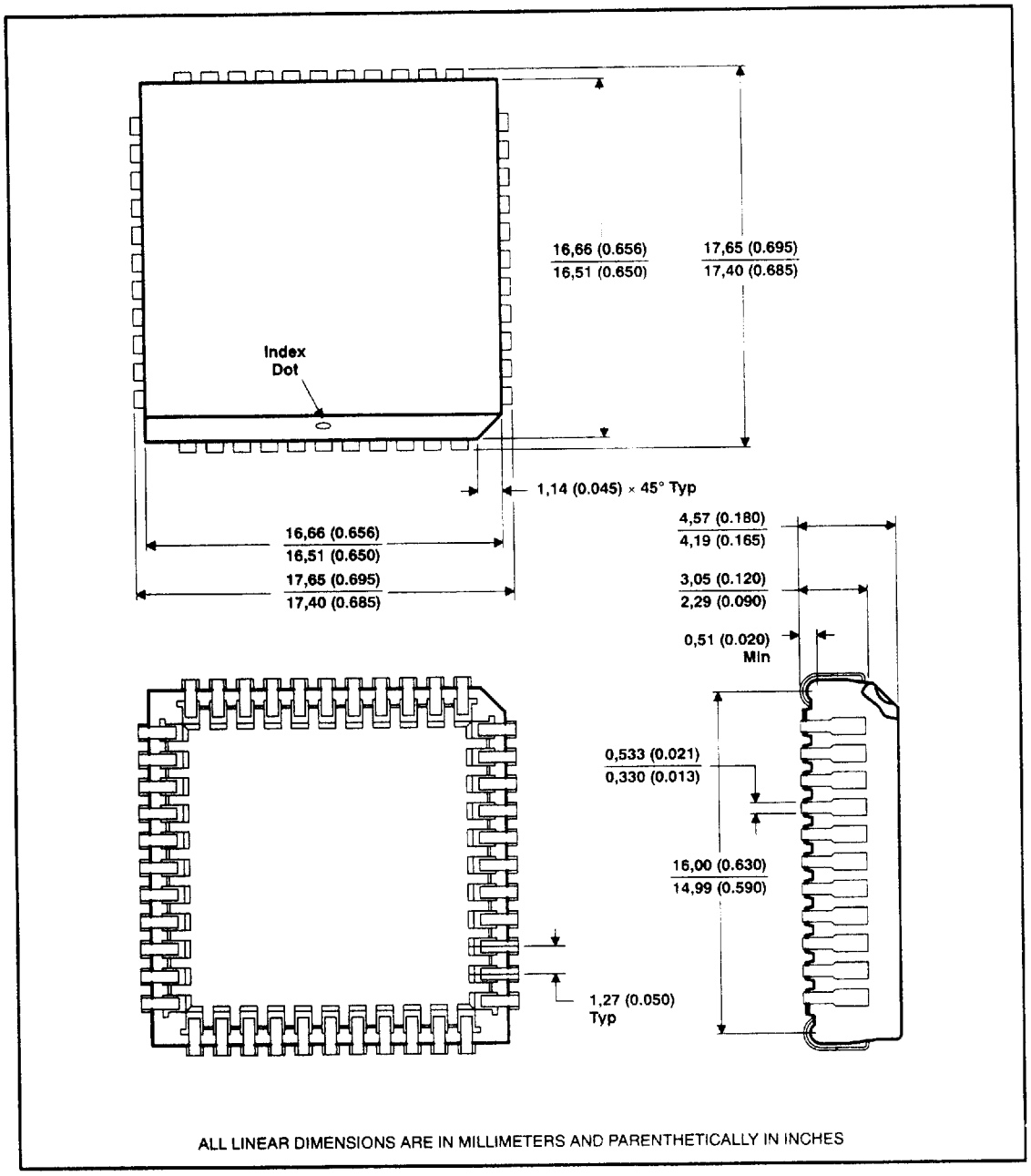
NOTE A: Each pin centerline is located within 0,254 (0.010) of its true longitudinal position.

40-pin windowed ceramic dual-in-line package



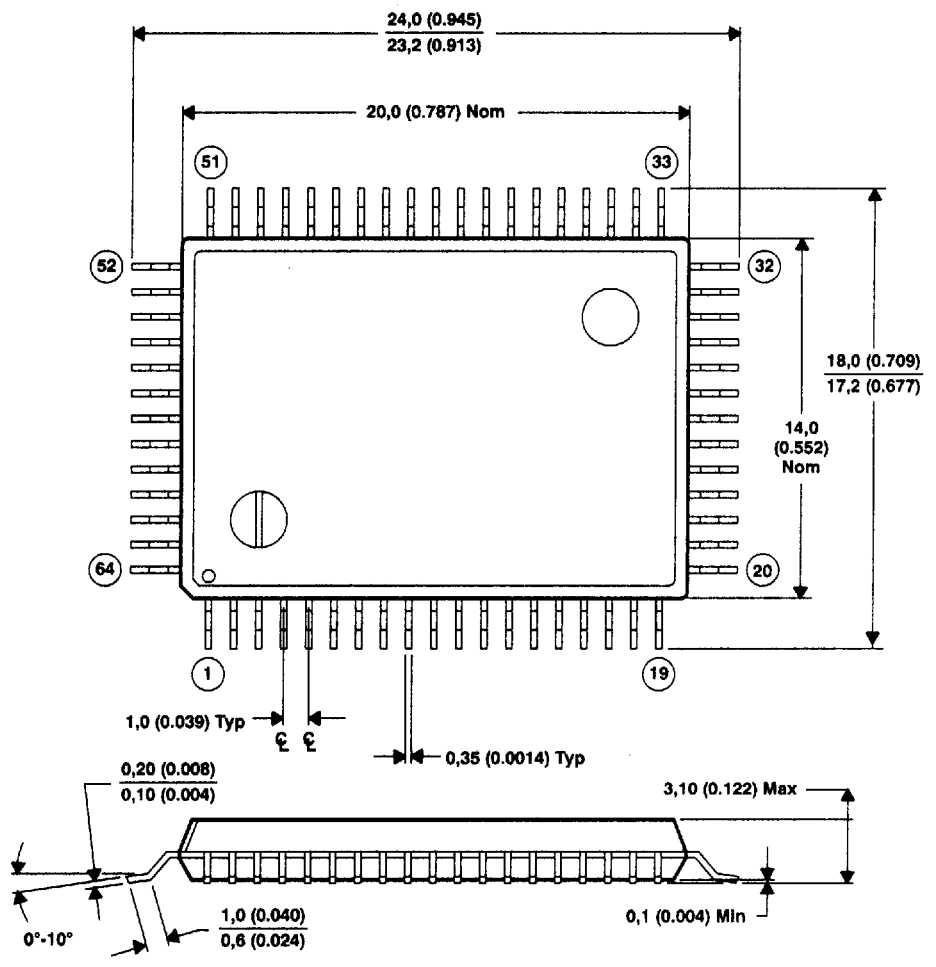
NOTE A: Each pin centerline is located within 0,254 (0.010) of its true longitudinal position.

44-lead plastic chip carrier (FN suffix)



ALL LINEAR DIMENSIONS ARE IN MILLIMETERS AND PARENTHETICALLY IN INCHES

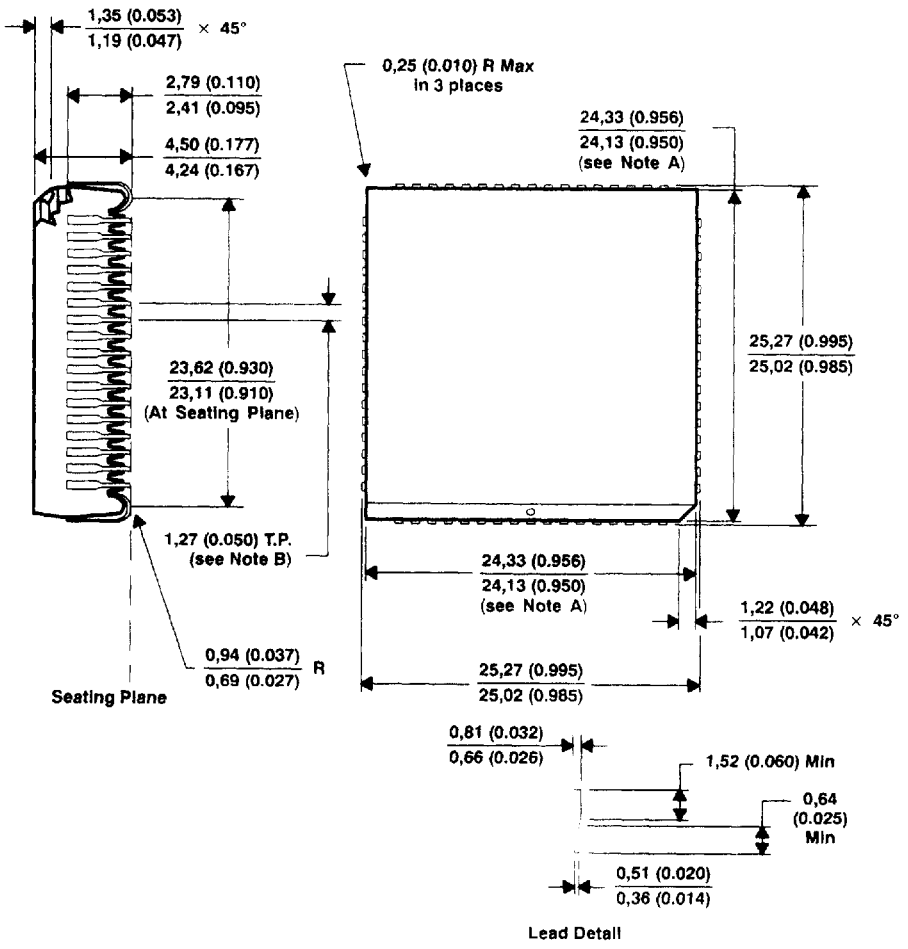
64-pin quad flat pack (PG suffix) (TMS320C16)



ALL LINEAR DIMENSIONS ARE IN MILLIMETERS AND PARENTHETICALLY IN INCHES

MECHANICAL DATA

68-lead plastic chip carrier package (FN suffix)

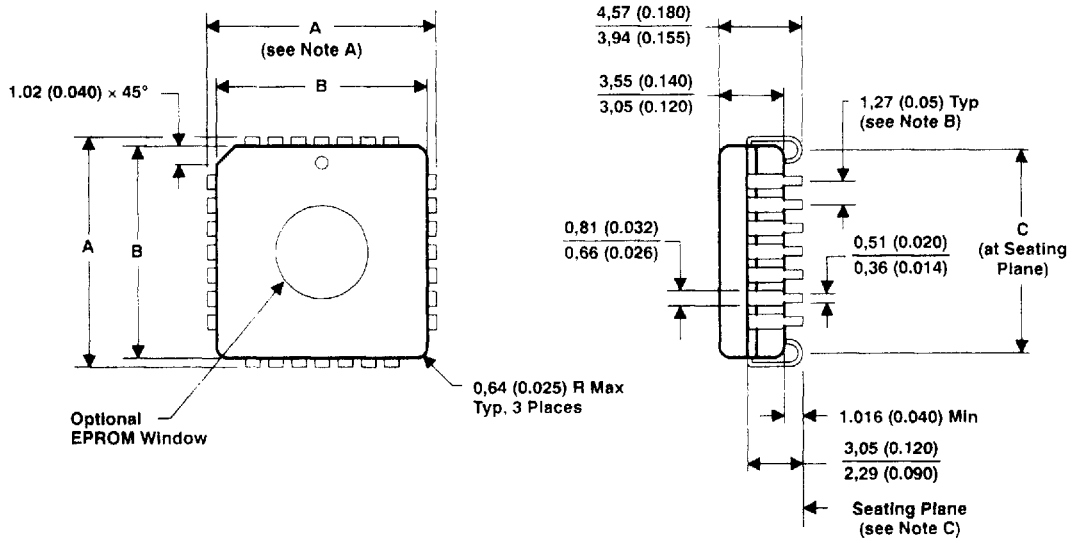


NOTES: A. Centerline of center pin, each side, is within 0,10 (0.004) of package centerline as determined by this deminsion.
B. Location of each pin is within 0,27 (0.005) of true position with respect to center pin on each side.

ALL LINEAR DIMENSIONS ARE IN MILLIMETERS AND PARENTHETICALLY IN INCHES

MECHANICAL DATA

68-lead ceramic chip carrier package (FZ suffix)



		A		B		C	
		MIN	MAX	MIN	MAX	MIN	MAX
M0-087AA	28	12,32 (0.485)	12,57 (0.495)	10,92 (0.430)	11,56 (0.455)	10,41 (0.410)	10,92 (0.430)
M0-087AB	44	17,40 (0.685)	17,65 (0.695)	16,00 (0.630)	16,64 (0.655)	15,49 0.610)	16,00 (0.630)
M0-087AD	68	25,02 (0.985)	25,27 (0.995)	23,62 (0.930)	24,26 (0.955)	23.11 (0.910)	23,62 (0.930)

NOTES: A. Centerline of center pin each side is within 0,10 (0.004) of package centerline as determined by dimension B.
B. Location of each pin is within 0,27 (0.005) of true position with respect to center pin on each side.
C. The lead contact points are planar within 0,15 (0.006)

ALL LINEAR DIMENSIONS ARE IN MILLIMETERS AND PARENTHETICALLY IN INCHES

Appendix B

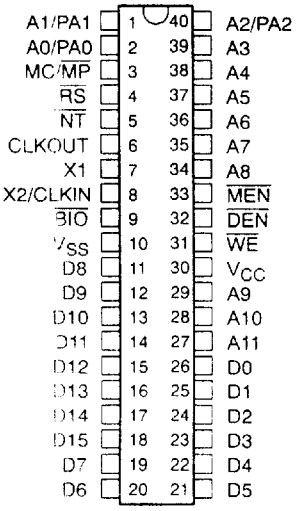
SMJ320C1x Digital Signal Processors

This appendix contains electrical specifications,timing, and mechanical data for all SMJ320C1x devices.

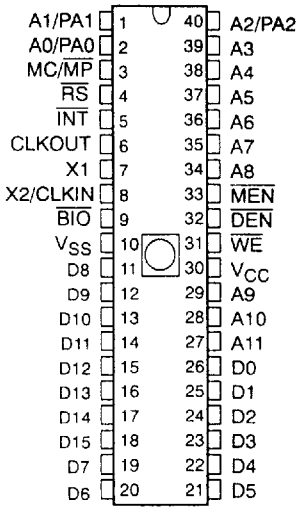
APRIL 1990

- 160/200-ns Instruction Cycle
- 144/256-Word On-Chip Data RAM
- 1.5K/4K-Word On-Chip Program ROM
- 4K-Word On-Chip Program EPROM (SMJ320E15)
- EPROM Code Protection for Copyright Security
- 4K-Word Total External Memory at Full Speed
- 32-Bit ALU/Accumulator
- 16 × 16-Bit Multiplier with a 32-Bit Product
- 0 to 16-Bit Barrel Shifter
- Eight Input and Eight Output Channels
- 16-Bit Bidirectional Data Bus with up to 50-Mbps Transfer Rate
- Single 5-V Power Supply
- Class B High Reliability Processing
- CMOS Technology:
  - SMJ320C10-14 . . . . . 280-ns cycle time
  - SMJ320C10 . . . . . 200-ns cycle time
  - SMJ320C10-25 . . . . . 160-ns cycle time
  - SMJ320C15 . . . . . 200-ns cycle time
  - SMJ320C15-25 . . . . . 160-ns cycle time
  - SMJ320E15 (EPROM) 200-ns cycle time

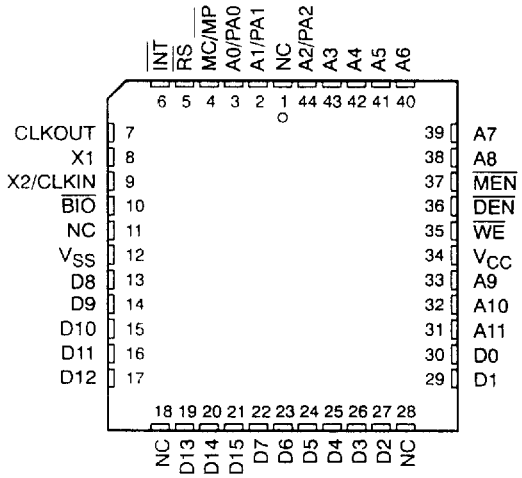
SMJ320C10, SMJ320C15
JD PACKAGE
(TOP VIEW)



SMJ320E15
WINDOWED JD PACKAGE
(TOP VIEW)



SMJ320C10, SMJ320C15
FD PACKAGE
(TOP VIEW)



This integrated SMJ320 digital signal processor family data sheet provides complete design documentation for all first-generation, high-reliability, CMOS DSP devices facilitating the selection of the devices best suited for user applications. Generic information is presented first, followed by specific device information. An index is provided for quick reference to specific information about a device

TEXAS INSTRUMENTS
POST OFFICE BOX 1443 • HOUSTON, TEXAS 77001

Introduction

The SMJ320 family of 16/32-bit single-chip CMOS Digital Signal Processors combines the flexibility of a high-speed controller with the numerical capability of an array processor, thereby offering an inexpensive alternative to multichip bit-slice processors. The SMJ320 family's unique versatility and power give the design engineer a new approach to a variety of complicated applications. In addition, these microcomputers are capable of providing the multiple functions often required for a single application. For example, the SMJ320 family can synthesize and recognize speech, sense objects with radar or optical intelligence, and perform mechanical operations through digital servo loop computations.

The SMJ320 family consists of three DSP generations. The first generation includes the SMJ320C10 and its spinoffs as described in this data sheet. The second generation includes the SMJ32020, the SMJ320C25 and the SMJ320E25 which are designed for higher performance. The third generation product is the SMJ320C30, the first floating point DSP. Its 60-ns cycle time allows execution of more than 33 million floating point operations per second (MegaFLOPS).

While many features are common throughout the SMJ320 DSP family, specific features are provided in each processor to provide different cost/performance tradeoffs. Software compatability is maintained throughout the SMJ320 family to protect the user's investment in architecture. Software and hardware tools are available for each DSP to facilitate rapid design

description

The SMJ320C10 Digital Signal Processor operates at 20 MHz, and has a 200-ns cycle time. Its CMOS process technology permits it to achieve a power dissipation of less than one-sixth that of an equivalent NMOS DSP. This significantly lower power dissipation makes the SMJ320C10 ideal for power-sensitive applications such as man-packed battery-backed radios

The SMJ320C10-14, a 14 MHz version, low power dissipation CMOS DSP, has a 280-ns instruction cycle time. It provides a low-cost alternative for DSP applications not requiring the maximum operating frequency of the SMJ320C10.

The CMOS SMJ320C10-25 operates at 25 MHz, has a 160-ns instruction cycle time, and is well suited for low power dissipation, high-performance DSP applications.

All versions of the SMJ320C10 DSP are object-code and pin-for-pin compatible with the SMJ32010 DSP.

The SMJ320C15, SMJ320C15-25, and the SMJ320E15 CMOS devices are object-code and pin-for-pin compatible with the SMJ32010 and offer expanded on-chip RAM of 256 words, and on-chip ROM or EPROM of 4K words. The SMJ320C15, a 20 MHz version, has a 200-ns cycle time. The SMJ320C15-25, a 25.6 MHz version, operates with a 160-ns cycle time.The SMJ320E15 (EPROM), a 20 MHz version, operates at a 200-ns cycle time. These devices within the CMOS SMJ320 DSP family allow the on-chip capability of upgrading performance while reducing power requirements, board space, and system cost without hardware or software redesign.

Table 1 provides an overview of the SMJ320 first generation DSPs within this data sheet. For specific availability, contact the nearest TI sales office

TABLE 1. SMJ320 FIRST-GENERATION DEVICE OVERVIEW

DEVICE	MEMORY				I/O PAR	CYCLE TIME (ns)	MAX POWER (mW)	PACKAGE TYPE	
	ON-CHIP			OFF-CHIP EXPANSION					
	RAM	ROM	EPROM					DIP	LCCC
SMJ320C10-14	144	1.5K		4K	8 × 16	280	275	40	44
SMJ320C10	144	1.5K		4K	8 × 16	200	275	40	44
SMJ320C10-25	144	1.5K		4K	8 × 16	160	330	40	44
SMJ320C15	256	4K		4K	8 × 16	200	275	40	44
SMJ320C15-25	256	4K		4K	8 × 16	160	330	40	44
SMJ320E15	256		4K	4K	8 × 16	200	495	40	–

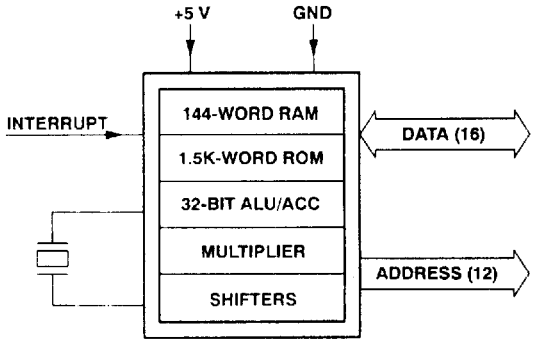
PIN NOMENCLATURE (SMJ320C10, SMJ320C15, SMJ320E15†)

NAME	I/O/Z‡	DEFINITION
A11-A0/PA2-PA0	O	External address bus. I/O port address multiplexed over PA2-PA0
$\overline{BIO}$	I	External polling input
CLKOUT	O	System clock output, 1/4 crystal/CLKIN frequency
D15-D0	I/O/Z	16-bit parallel data bus
$\overline{DEN}$	O	Data enable for device input data on D15-D0
$\overline{INT}$	I	External interrupt input
MC/$\overline{MP}$	I	Memory mode select pin. High selects microcomputer mode. Low selects microprocessor mode.
$\overline{MEN}$	O	Memory enable indicates that D15-D0 will accept external memory instruction.
NC	—	No connection; make no external connection to this pin.
$\overline{RS}$	I	Reset for initializing the device
V_{CC}	I	+ 5 V supply
V_{SS}	I	Ground
$\overline{WE}$	O	Write enable for device output data on D15-D0
X1	O	Crystal output for internal oscillator
X2/CLKIN	I	Crystal input for internal oscillator or external system clock input

† See EPROM programming section.
‡ Input/Output/High-impedance state.

Key Features: SMJ320C10

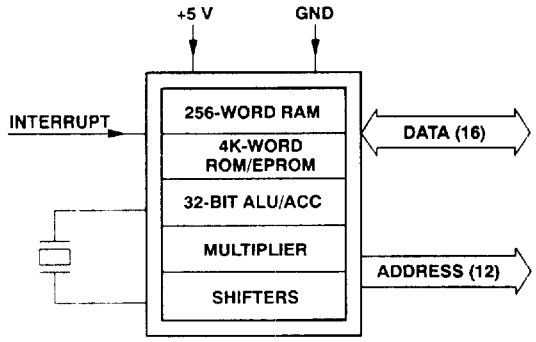
- Instruction Cycle Timing:
  — 160-ns (SMJ320C10-25)
  — 200-ns (SMJ320C10)
  — 280-ns (SMJ320C10-14)
- 144 Words of On-Chip Data RAM
- 1.5K Words of On-Chip Program ROM
- External Memory Expansion up to 4K Words at Full Speed
- 16 × 16-Bit Multiply in One Instruction Cycle
- 0 to 16-Bit Barrel Shifter
- Object Code and Pin-for-Pin Compatible with SMJ32010
- On-Chip Clock Oscillator
- Single 5-V Power Supply
- Device Packaging:
  — 40-Pin Side-Brazed Ceramic DIP
  — 44-Pad Leadless Ceramic Chip Carrier



- 16-Bit Instruction Data Word
- 32-Bit ALU/Accumulator
- CMOS Technology
- Class B Processing

Key Features: SMJ320C15/E15

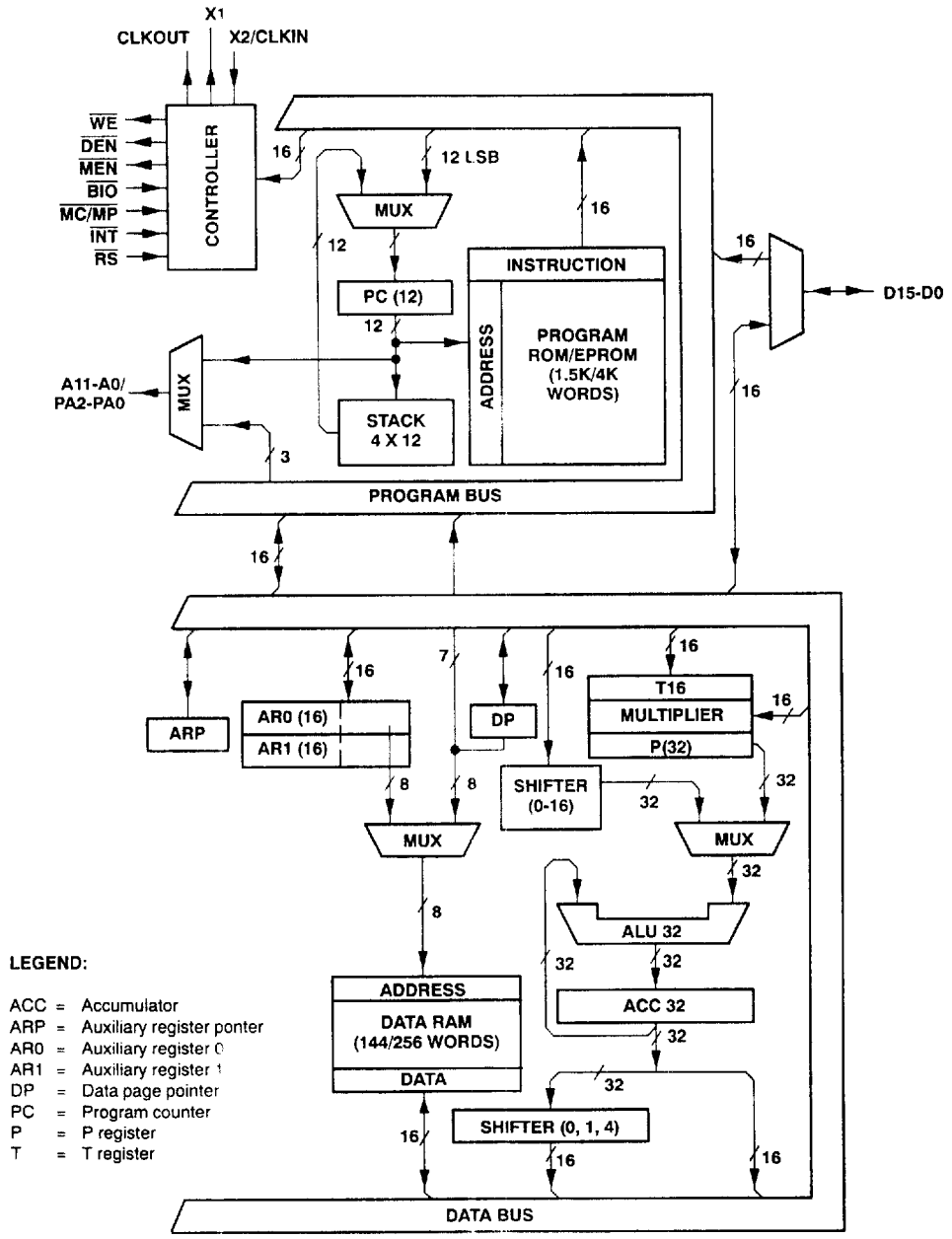
- Instruction Cycle Timing:
  — 160-ns (SMJ320C15-25)
  — 200-ns (SMJ320C15/E15)
- 256 Words of On-Chip Data RAM
- 4K Words of On-Chip Program ROM (SMJ320C15/C15-25)
- 4K Words of On-Chip Program EPROM (SMJ320E15)
- External Memory Expansion up to 4K Words at Full Speed
- EPROM Code Protection for Copyright Security
- Object Code and Pin-for-Pin Compatible with SMJ32010
- 16 × 16-Bit Multipler with 32-Bit Product
- 0 to 16-Bit Barrel Shifter
- On-Chip Clock Oscillator
- Single 5-V Power Supply
- CMOS Technology
- Class B Processing



- Device Packaging:
  — 40-Pin Side-Brazed Ceramic DIP (SMJ320C15, SMJ320C15-25)
  — 40-Pin Side Brazed Windowed Ceramic DIP (SMJ320E15)
  — 44-Pad Leadless Ceramic Chip Carrier (SMJ320C15)

TEXAS INSTRUMENTS
POST OFFICE BOX 1443 • HOUSTON, TEXAS 77001

functional block diagram (SMJ320C10, SMJ320C15, SMJ320E15)



LEGEND:

ACC = Accumulator
ARP = Auxiliary register ponter
AR0 = Auxiliary register 0
AR1 = Auxiliary register 1
DP = Data page pointer
PC = Program counter
P = P register
T = T register

architecture

The SMJ320 family utilizes a modified Harvard architecture for speed and flexibility. In a strict Harvard architecture, program and data memory lie in two separate spaces, permitting a full overlap of instruction fetch and execution. The SMJ320 family's modification of the Harvard architecture allows transfers between program and data spaces, thereby increasing the flexibility of the device. This modification permits coefficients stored in program memory to be read into the RAM, eliminating the need for a separated coefficient ROM. It also makes available immediate instructions and subroutines based on computed values.

32-bit ALU/accumulator

The SMJ320 first-generation devices contain a 32-bit ALU and accumulator for support of double-precision, two's-complement arithmetic. The ALU is a general-purpose arithmetic unit that operates on 16-bit words taken from the data RAM or derived from immediate instructions. In addition to the usual arithmetic instructions, the ALU can perform Boolean operations, providing the bit manipulation ability required of a high-speed controller. The accumulator stores the output from the ALU and is often an input to the ALU. It operates with a 32-bit wordlength. The accumulator is divided into a high-order word (bits 31 through 16) and a low-order word (bits 15 through 0). Instructions are provided for storing the high- and low-order accumulator words in memory.

shifters

Two shifters are available for manipulating data. The ALU barrel shifter performs a left-shift of 0 to 16 places on data memory words loaded into the ALU. This shifter extends the high-order bit of the data word and zero-fills the low-order bits for two's-complement arithmetic. The accumulator parallel shifter performs a left-shift of 0, 1, or 4 places on the entire accumulator and places the resulting high-order accumulator bits into data RAM. Both shifters are useful for scaling and bit extraction.

16 × 16-bit parallel multiplier

The multiplier performs a 16 × 16-bit two's-complement multiplication with a 32-bit result in a single instruction cycle. The multiplier consists of three units: the T Register, P Register, and multiplier array. The 16-bit T Register temporarily stores the multiplicand; the P Register stores the 32-bit product. Multiplier values either come from the data memory or are derived immediately from the MPYK (multiply immediate) instruction word. The fast on-chip multiplier allows the device to perform fundamental operations such as convolution, correlation, and filtering.

data and program memory

Since the SMJ320 devices use a Harvard architecture, data and program memory reside in two separate spaces. The first-generation devices have 144 or 256 words of on-chip data RAM and 1.5K or 4K words of on-chip program ROM. On-chip program EPROM of 4K words is provided on the SMJ320E15. The EPROM cell utilizes standard PROM programmers and is programmed identically to a 64K CMOS EPROM (SMJ27C64).

program memory expansion

The first-generation devices are capable of executing up to 4K words of external memory at full speed for those applications requiring external program memory space. This allows for external RAM-based systems to provide multiple functionality.

interrupts and subroutines

The SMJ320 first-generation devices contain a four-level hardware stack for saving the contents of the program counter during interrupts and subroutine calls. Instructions are available for saving the device's complete context. PUSH or POP instructions permit a level of nesting restricted only by the amount of available RAM. The interrupts used in these devices are maskable.

TEXAS INSTRUMENTS
POST OFFICE BOX 1443 • HOUSTON, TEXAS 77001

microcomputer/microprocessor operating modes (SMJ320C10/C15)

The SMJ320C10 and SMJ320C15 devices offer two modes of operation defined by the state of the MC/$\overline{MP}$ pin: the microcomputer mode (MC/$\overline{MP}$ = 1) or the microprocessor mode (MC/$\overline{MP}$ = 0). In the microcomputer mode, on-chip ROM is mapped into the memory space with up to 4K words of memory available. In the microprocessor mode, all 4K words of memory are external.

input/output

The 16-bit parallel data bus can be utilized to perform I/O functions in two cycles. The I/O ports are addressed by the three LSBs on the address lines. In addition, a polling input for bit test and jump operations ($\overline{BIO}$) and an interrupt pin ($\overline{INT}$) have been incorporated for multitasking.

instruction set

A comprehensive instruction set supports both numeric-intensive operations, such as signal processing, and general-purpose operations, such as high-speed control. All of the first-generation devices are object-code compatible and use the same 60 instructions. The instruction set consists primarily of single-cycle single-word instructions, permitting execution rates of more than six million instructions per second. Only infrequently used branch and I/O instructions are multicycle. Instructions that shift data as part of an arithmetic operation execute in a single cycle and are useful for scaling data in parallel with other operations.

Three main addressing modes are available with the instruction set: direct, indirect, and immediate addressing.

direct addressing

In direct addressing, seven bits of the instruction word concatenated with the 1-bit data page pointer form the data memory address. This implements a paging scheme in which the first page contains 128 words, and the second page contains up to 128 words.

indirect addressing

Indirect addressing forms the data memory address from the least-significant eight bits of one of the two auxiliary registers, AR0-AR1. The Auxiliary Register Pointer (ARP) selects the current auxiliary register. The auxiliary registers can be automatically incremented or decremented and the ARP changed in parallel with the execution of any indirect instruction to permit single-cycle manipulation of data tables. Indirect addressing can be used with all instructions requiring data operands, except for the immediate operand instructions.

immediate addressing

Immediate instructions derive data from part of the instruction word rather than from the data RAM. Some useful immediate instructions are multiply immediate (MPYK), load accumulator immediate (LACK), and load auxiliary register immediate (LARK).

instruction set summary

Table 2 lists the symbols and abbreviations used in Table 3, the instruction set summary. Table 3 contains a short description and the opcode for each SMJ320 first-generation instruction. The summary is arranged according to function and alphabetized within each functional group.

TABLE 2. INSTRUCTION SYMBOLS

SYMBOL	MEANING
ACC	Accumulator
D	Data memory address field
I	Addressing mode bit
K	Immediate operand field
PA	3-bit port address field
R	1-bit operand field specifying auxiliary register
S	4-bit left-shift code
X	3-bit accumulator left-shift field

TABLE 3. SMJ320 FIRST-GENERATION INSTRUCTION SET SUMMARY

ACCUMULATOR INSTRUCTIONS

MNEMONIC	DESCRIPTION	NO. CYCLES	NO. WORDS	OPCODE INSTRUCTION REGISTER															
				15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ABS	Absolute value of accumulator	1	1	0	1	1	1	1	1	1	1	I	0	0	0	1	0	0	0
ADD	Add to accumulator with shift	1	1	0	0	0	0	←		S	→	I	←			D			→
ADDH	Add to high-order accumulator bits	1	1	0	1	1	0	0	0	0	0	I	←			D			→
ADDS	Add to accumulator with no sign extension	1	1	0	1	1	0	0	0	0	1	I	←			D			→
AND	AND with accumulator	1	1	0	1	1	1	1	0	0	1	I	←			D			→
LAC	Load accumulator with shift	1	1	0	0	1	0	←		S	→	I	←			D			→
LACK	Load accumulator immediate	1	1	0	1	1	1	1	1	1	0	←				K			→
OR	OR with accumulator	1	1	0	1	1	1	1	0	1	0	I	←			D			→
SACH	Store high-order accumulator bits with shift	1	1	0	1	0	1	1	←	X	→	I	←			D			→
SACL	Store low-order accumnulator bits	1	1	0	1	0	1	0	0	0	0	I	←			D			→
SUB	Subtract from accumulator with shift	1	1	0	0	0	1	←		S	→	I	←			D			→
SUBC	Conditional subtract (for divide)	1	1	0	1	1	0	0	1	0	0	I	←			D			→
SUBH	Subtract from high-order accumulator bits	1	1	0	1	1	0	0	0	1	0	I	←			D			→
SUBS	Subtract from accumulator with no sign extension	1	1	0	1	1	0	0	0	1	1	I	←			D			→
XOR	Exclusive OR with accumulator	1	1	0	1	1	1	1	0	0	0	I	←			D			→
ZAC	Zero accumulator	1	1	0	1	1	1	1	1	1	1	I	0	0	0	1	0	0	1
ZALH	Zero accumulator and load high-order bits	1	1	0	1	1	0	0	1	0	1	I	←			D			→
ZALS	Zero accumulator and load low-order bits with no sign extension	1	1	0	1	1	0	0	1	1	0	I	←			D			→

AUXILIARY REGISTER AND DATA PAGE POINTER INSTRUCTIONS

MNEMONIC	DESCRIPTION	NO. CYCLES	NO. WORDS	OPCODE INSTRUCTION REGISTER															
				15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
LAR	Load auxiliary register	1	1	0	0	1	1	1	0	0	R	I	←			D			→
LARK	Load auxiliary register immediate	1	1	0	1	1	1	0	0	0	R	←				K			→
LARP	Load auxiliary register pointer immediate	1	1	0	1	1	0	1	0	0	0	I	0	0	0	0	0	0	K
LDP	Load data memory page pointer	1	1	0	1	1	0	1	1	1	1	I	←			D			→
LDPK	Load data memory page pointer immediate	1	1	0	1	1	0	1	1	1	0	0	0	0	0	0	0	0	K
MAR	Modify auxiliary register and pointer	1	1	0	1	1	0	1	0	0	0	I	←			D			→
SAR	Store auxiliary register	1	1	0	0	1	1	0	0	0	R	I	←			D			→

TEXAS INSTRUMENTS
POST OFFICE BOX 1443 • HOUSTON, TEXAS 77001

TABLE 3. SMJ320 FIRST-GENERATION INSTRUCTION SET SUMMARY (continued)

BRANCH INSTRUCTIONS

MNEMONIC	DESCRIPTION	NO. CYCLES	NO. WORDS	OPCODE INSTRUCTION REGISTER 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
B	Branch unconditionally	2	2	1 1 1 1 1 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 ← BRANCH ADDRESS →
BANZ	Branch on auxiliary register not zero	2	2	1 1 1 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 ← BRANCH ADDRESS →
BGEZ	Branch if accumulator ≥ 0	2	2	1 1 1 1 1 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 ← BRANCH ADDRESS →
BGZ	Branch if accumulator > 0	2	2	1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 ← BRANCH ADDRESS →
BIOZ	Branch on $\overline{BIO}$ = 0	2	2	1 1 1 1 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 ← BRANCH ADDRESS →
BLEZ	Branch if accumulator ≤ 0	2	2	1 1 1 1 1 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0 ← BRANCH ADDRESS →
BLZ	Branch if accumulator < 0	2	2	1 1 1 1 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 ← BRANCH ADDRESS →
BNZ	Branch if accumulator ≠ 0	2	2	1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 ← BRANCH ADDRESS →
BV	Branch on overflow	2	2	1 1 1 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 ← BRANCH ADDRESS →
BZ	Branch if accumulator = 0	2	2	1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 ← BRANCH ADDRESS →
CALA	Call subroutine from accumulator	2	1	0 1 1 1 1 1 1 1 1 0 0 0 1 1 0 0
CALL	Call subroutine immediately	2	2	1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 ← BRANCH ADDRESS →
RET	Return from subroutine or interrupt routine	2	1	0 1 1 1 1 1 1 1 1 0 0 0 1 1 0 1

T REGISTER, P REGISTER, AND MULTIPLY INSTRUCTIONS

MNEMONIC	DESCRIPTION	NO. CYCLES	NO. WORDS	OPCODE INSTRUCTION REGISTER 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
APAC	Add P register to accumulator	1	1	0 1 1 1 1 1 1 1 1 0 0 0 1 1 1 1
LT	Load T Register	1	1	0 1 1 0 1 0 1 0 I ← D →
LTA	LTA combines LT and APAC into one instruction	1	1	0 1 1 0 1 1 0 0 I ← D →
LTD	LTD combines LT, APAC, and DMOV into one instruction	1	1	0 1 1 0 1 0 1 1 I ← D →
MPY	Multiply with T register, store product in P register	1	1	0 1 1 0 1 1 0 1 I ← D →
MPYK	Multiply T register with immediate operand; store product in P register	1	1	1 0 0 ← K →
PAC	Load accumulator from P register	1	1	0 1 1 1 1 1 1 1 1 0 0 0 1 1 1 0
SPAC	Subtract P register from accumulator	1	1	0 1 1 1 1 1 1 1 1 0 0 1 0 0 0 0

TABLE 3. SMJ320 FIRST-GENERATION INSTRUCTION SET SUMMARY (concluded)

CONTROL INSTRUCTIONS																		
MNEMONIC	DESCRIPTION	NO. CYCLES	NO. WORDS	OPCODE INSTRUCTION REGISTER														
				15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DINT	Disable interrupt	1	1	0	1	1	1	1	1	1	1	I	0	0	0	0	0	0	1
EINT	Enable interrupt	1	1	0	1	1	1	1	1	1	1	I	0	0	0	0	0	1	0
LST	Load status register	1	1	0	1	1	1	1	0	1	1	I	◄				D		►
NOP	No operation	1	1	0	1	1	1	1	1	1	1	I	0	0	0	0	0	0	0
POP	POP stack to accumulator	2	1	0	1	1	1	1	1	1	1	I	0	0	1	1	1	0	1
PUSH	PUSH stack from accumulator	2	1	0	1	1	1	1	1	1	1	I	0	0	1	1	1	0	0
ROVM	Reset overflow mode	1	1	0	1	1	1	1	1	1	1	I	0	0	0	1	0	1	0
SOVM	Set overflow mode	1	1	0	1	1	1	1	1	1	1	I	0	0	0	1	0	1	1
SST	Store status register	1	1	0	1	1	1	1	1	0	0	I	◄				D		►
I/O AND DATA MEMORY OPERATIONS																			
MNEMONIC	DESCRIPTION	NO. CYCLES	NO. WORDS	OPCODE INSTRUCTION REGISTER															
				15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DMOV	Copy contents of data memory location into next higher location	1	1	0	1	1	0	1	0	0	1	I	◄				D		►
IN	Input data from port	2	1	0	1	0	0	0	◄	PA	►	I	◄				D		►
OUT	Output data to port	2	1	0	1	0	0	1	◄	PA	►	I	◄				D		►
TBLR	Table read from program memory to data RAM	3	1	0	1	1	0	0	1	1	1	I	◄				D		►
TBLW	Table write from data RAM to program memory	3	1	0	1	1	1	1	1	0	1	I	◄				D		►

development support products

Texas Instruments offers an extensive line of development products to assist the user in all aspects of SMJ320 first-generation-based design and development. These products range from development and application software to complete hardware development and evaluation systems such as the XDS/22. Table 4 lists the development support products for the first-generation SMJ320 devices.

System development begins with the use of the Evaluation Module (EVM) or Emulator (XDS). These hardware tools allow the designer to evaluate the processor's performance, benchmark time-critical code, and determine the feasibility of using a SMJ320 device to implement a specific algorithm.

Software and hardware can be developed in parallel by using the macro assembler/linker and simulator for software development and the XDS for hardware development. The assembler/linker translates the system's assembly source program into an object module that can be executed by the simulator, XDS, or EVM. The XDS provides realtime in-circuit emulation and is a powerful tool for debugging and integrating software and hardware modules.

Additional support for the SMJ320 products consists of an extensive library of product and applications documentation. Three-day DSP design workshops are offered by the TI Regional Technology Centers (RTCs). These workshops provide insight into the architecture and the instruction set of the first-generation SMJ320s as well as hands-on training with the SMJ320 development tools. When technical questions arise in regard to a SMJ320 member, contact Texas Instruments TMS320 Hotline (713) 274-2320. Or, keep informed on the latest TI and third-party development support tools by accessing the libraries of application source code via the DSP Bulletin Board Service (BBS) at (713) 274-2323. The BBS provides access for the 2400-/1200-/300-bps modems.

TABLE 4. TMS320 FIRST-GENERATION SOFTWARE AND HARDWARE SUPPORT

SOFTWARE TOOLS	PART NUMBER
Macro Assembler/Linker	
PC/MS-DOS	TMDS3242850-02
VAX/VMS	TMDS3242250-08
VAX ULTRIX	TMDS3242260-08
SUN-3 UNIX	TMDS3242550-08
Simulator	
PC/MS-DOS	TMDS3240811-02
VAX/VMS	TMDS3240211-08
Digital Filter Design Package (DFDP)	
IBM PC PC-DOS	DFDP/IBM002
DSP Software Software Library	
PC/MS-DOS	TMDC3240812-12
VAX/VMS	TMDC3240212-18
TMS320 Bell 212A Modem Software	
PC/MS-MOS	TMDX3240813-12
Data Encryption Standard Software	
PC/MS-DOS	TMDX3240814-12
HARDWARE TOOLS	**PART NUMBER**
Evaluation Tools	
Evaluation Module (EVM)	RTC/EVM320A-03
Analog Interface Board 1 (AIB1)	RTC/EVM320C-06
Analog Interface Board 2 (AIB2)	RTC/AIB320A-06
XDS/22 Emulators	
TMS320C10/C15	TMDS3262211
XDS/22 Upgrade Kits	
TMS32010 → TMS320C10/C15	TMDS3282215
EPROM Programming Adaptor Sockets	
40- to 28-pin (TMS320E15)	RTC/PGM320A-06
Additional Target Connector	
44-pin (TMS320C10/C15)	TMDX3288810

documentation support

Extensive documentation supports the first-generation SMJ320 devices from product announcement through applications development. The types of documentation include data sheets with design specifications, complete user's guides, and 750 pages of application reports published in the book *Digital Signal Processing Applications with the TMS320 Family*.

A series of DSP textbooks is being published to support digital signal processing research and education. The first book, *DFT/FFT and Convolution Algorithms*, is now available. The TMS320 newsletter, *Details on Signal Processing*, is published quarterly and distributed to update TMS320 customers on product information. The TMS320 DSP bulletin board service board service provides access to large amounts of information pertaining to the TMS320 family.

Refer to the *TMS320 Family Development Support Reference Guide* for further information about TMS320 documentation. To receive copies of first-generation SMJ320 literature, call the Customer Response Center at 1-800-232-2300.

DEVICE ELECTRICAL SPECIFICATIONS

This section contains all the electrical specifications for the SMJ320 CMOS first-generation devices. Refer to the top corner for the specific device.

absolute maximum ratings over operating free-air temperature range (unless otherwise noted)†

Supply voltage range (see Note 1) . . . . . – 0.3 V to 7 V
Input voltage range : SMJ320C10/C10-14/C10-25/C15/C15-25 . . . . . – 0.3 V to 7 V
SMJ320E15 . . . . . – 0.3 V to 15 V
Output voltage range . . . . . – 0.3 V to 7 V
Continuous power dissipation: SMJ320C10/C15 . . . . . 275 mW
SMJ320C10-14 . . . . . 275 mW
SMJ320C10-25/C15-25 . . . . . 330 mW
SMJ320E15 . . . . . 495 mW
Maximum operating case temperature . . . . . 125 °C
Minimum operating free-air temperature . . . . . – 55 °C
Storage temperature . . . . . – 65 °C to 150 °C

† Stresses beyond those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. This is a stress rating only, and functional operation of the device at these or any other conditions beyond those indicated in the "Recommended Operating Conditions" section of this specification is not implied. Exposure to absolute-maximum-rated conditions for extended periods may affect device reliability.

NOTE 1: Under "Absolute Maximum Ratings", all voltage values are with respect to V_{SS}.

recommended operating conditions

			MIN	NOM	MAX	UNIT
V_{CC}	Supply voltage		4.5	5	5.5	V
V_{SS}	Supply voltage			0		V
V_{IH}	High-level input voltage	All inputs except CLKIN	2			V
		CLKIN	3			
V_{IL}	Low-level input voltage	All inputs except MC/$\overline{MP}$			0.8	V
		MC/$\overline{MP}$			0.6	
I_{OH}	High-level output current (all outputs)				– 300	μA
I_{OL}	Low-level output current (all outputs)				2	mA
T_A	Operating free-air temperature		– 55			°C
T_C	Operating case temperature				125	°C

electrical characteristics over specified temperature range (unless otherwise noted)

PARAMETER			TEST CONDITIONS		MIN	TYP‡	MAX	UNIT
V_{OH}	High-level output voltage		I_{OH} = MAX		2.4	3		V
			I_{OH} = 20 μA (see Note 2)		V_{CC} – 0.4‡			
V_{OL}	Low-level output voltage		I_{OL} = MAX			0.3	0.5	V
I_{OZ}	Off-state output current		V_{CC} = MAX	V_O = 2.4 V			20	μA
				V_O = 0.4 V			– 20	
I_I	Input current		V_{CC} = V_{SS} to V_{CC}	All inputs except CLKIN			±20	μA
				CLKIN			±50	
C_i	Input capacitance§	Data bus	f = 1 MHz, All other pins 0 V			25		pF
		All others				15		
C_o	Output capacitance§	Data bus				25		pF
		All others				10		

‡ All typical values are at V_{CC} = 5 V, T_A = 25°C.

§ This parameter is not production tested.

NOTE 2: This voltage specification is included for interface to HC logic. However, note that all of the other timing parameters defined in this data sheet are specified for TTL logic levels and will differ for HC logic levels.

TEXAS INSTRUMENTS
POST OFFICE BOX 1443 • HOUSTON, TEXAS 77001

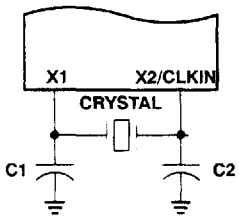


FIGURE 1. INTERNAL CLOCK OPTION

PARAMETER MEASUREMENT INFORMATION

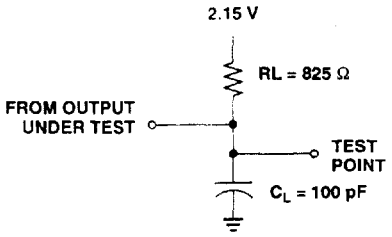


FIGURE 2. TEST LOAD CIRCUIT

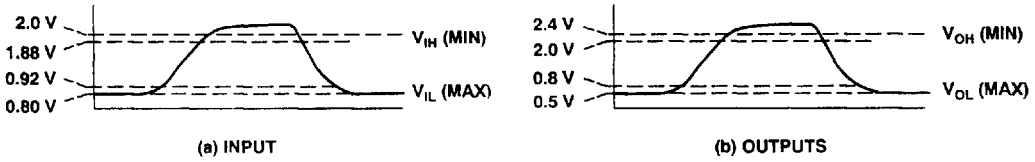


FIGURE 3. VOLTAGE REFERENCE LEVELS

electrical characteristics over specified termperature range (unless otherwise noted)

PARAMETER		TEST CONDITIONS (SEE FIGURE 2)	MIN	TYP†	MAX	UNIT
I_{CC}‡ Supply current	SMJ320C10-14	f = 14.4 MHz, V_{CC} = 5.5 V			50	mA
	SMJ320C10	f = 20.5 MHz, V_{CC} = 5.5 V			50	
	SMJ320C10-25	f = 25.6 MHz, V_{CC} = 5.5 V			60	

† All typical values are at V_{CC} = 5 V, T_A = 25°C.
‡ I_{CC} characteristics are inversely proportional to temperature, i.e., I_{CC} decreases approximately linearly with temperature.

CLOCK CHARACTERISTICS AND TIMING

The SMJ320C10 can use either its internal oscillator or an external frequency source for a clock.

internal clock option

The internal oscillator is enabled by connecting a crystal across X1 and X2/CLKIN (see Figure 1). The frequency of CLKOUT is one-fourth the crystal fundamental frequency. The crystal should be fundamental mode, and parallel resonant, with an effective series resistance of 30 ohms, a power dissipation of 1 mW, and should be specified at a load capacitance of 20 pF.

PARAMETER		TEST CONDITIONS	MIN	NOM	MAX	UNIT
Crystal frequency, f_x¶	SMJ320C10-14	–55°C to 125°C with 8 MHz crystal	6.7		14.4	MHz
	SMJ320C10	–55°C to 125°C with 8 MHz crystal	6.7		20.5	
	SMJ320C10-25	–55°C to 125°C with 8 MHz crystal	6.7		25.6	
C1, C2¶		–55°C to 125°C with 8 MHz crystal		10		pF

external clock option

An external frequency source can be used by injecting the frequency directly into X2/CLKIN with X1 left unconnected. The external frequency injected must conform to the specifications listed in the table below.

switching characteristics over recommended operating conditions

PARAMETER		TEST CONDITIONS (SEE FIGURE 2)	SMJ320C10-14			SMJ320C10			SMJ320C10-25			UNIT
			MIN	NOM	MAX	MIN	NOM	MAX	MIN	NOM	MAX	
$t_{c(C)}$	CLKOUT cycle time§	R_L = 825 Ω, C_L = 100 pF	277		600	195.12		600	156.25		600	ns
$t_{r(C)}$	CLKOUT rise time			10			10			10		ns
$t_{f(C)}$	CLKOUT fall time			8			8			8		ns
$t_{w(CL)}$	Pulse duration, CLKOUT low			92			92			72		ns
$t_{w(CH)}$	Pulse duration, CLKOUT high			90			90			70		ns
$t_{d(MCC)}$¶	Delay time, CLKIN↑ to CLKOUT↓		15		40	15		40	15		40	ns

§ $t_{c(C)}$ is the cycle time of CLKOUT, i.e., $4 \times t_{c(MC)}$ (4 times CLKIN cycle time if an external oscillator is used).
¶ This parameter is not production tested.

timing requirements over recommended operating conditions (unless otherwise noted) (see Note 3)

		SMJ320C10-14			SMJ320C10			SMJ320C10-25			UNIT
		MIN	NOM	MAX	MIN	NOM	MAX	MIN	NOM	MAX	
$t_{c(MC)}$	Master clock cycle time	69.44	50	150	48.78	50	150	39.06	40	150	ns
$t_{r(MC)}$	Rise time, master clock input		5			5			5		ns
$t_{f(MC)}$	Fall time, master clock input		5			5			5		ns
$t_{w(MCP)}$†	Pulse duration, master clock	$0.4t_{c(MC)}$		$0.6t_{c(MC)}$	$0.4t_{c(MC)}$		$0.6t_{c(MC)}$	$0.45t_{c(MC)}$		$0.55t_{c(MC)}$	ns
$t_{w(MCL)}$	Pulse duration, master clock low		20			20			15		ns
$t_{w(MCH)}$	Pulse duration, master clock high		20			20			15		ns

† This parameter is not production tested.
NOTE 3: CLKIN rise and fall times must be less than 10 ns.

MEMORY AND PERIPHERAL INTERFACE TIMING

switching characteristics over recommended operating conditions

PARAMETER		TEST CONDITIONS (SEE FIGURE 2)	SMJ320C10-14 SMJ320C10			SMJ320C10-25			UNIT
			MIN	NOM	MAX	MIN	NOM	MAX	
t_{d1}	Delay time, CLKOUT↓ to address bus valid	R_L = 825 Ω, C_L = 100 pF	10‡		50	10‡		40	ns
t_{d2}	Delay time, CLKOUT↓ to $\overline{MEN}$↓		$^1/_4t_{c(C)} - 5$‡		$^1/_4t_{c(C)} + 15$	$^1/_4t_{c(C)} - 5$‡		$^1/_4t_{c(C)} + 12$	ns
t_{d3}	Delay time, CLKOUT↓ to $\overline{MEN}$↑		−10‡		15	−10‡		12	ns
t_{d4}	Delay time, CLKOUT↓ to $\overline{DEN}$↓		$^1/_4t_{c(C)} - 5$‡		$^1/_4t_{c(C)} + 15$	$^1/_4t_{c(C)} - 5$‡		$^1/_4t_{c(C)} + 12$	ns
t_{d5}	Delay time, CLKOUT↓ to $\overline{DEN}$↑		−10‡		15	−10‡		12	ns
t_{d6}	Delay time, CLKOUT↓ to $\overline{WE}$↓		$^1/_2t_{c(C)} - 5$‡		$^1/_2t_{c(C)} + 15$	$^1/_2t_{c(C)} - 5$‡		$^1/_2t_{c(C)} + 12$	ns
t_{d7}	Delay time, CLKOUT↓ to $\overline{WE}$↑		−10‡		15	−10‡		12	ns
t_{d8}	Delay time, CLKOUT↓ to data bus OUT valid				$^1/_4t_{c(C)} + 65$			$^1/_4t_{c(C)} + 52$	ns
t_{d9}	Time after CLKOUT↓ that data bus starts to be driven		$^1/_4t_{c(C)} - 5$‡			$^1/_4t_{c(C)} - 5$‡			ns
t_{d10}	Time after CLKOUT↓ that data bus stops bieng driven				$^1/_4t_{c(C)+40}$‡			$^1/_4t_{c(C)+40}$‡	ns
t_v	Data bus OUT valid after CLKOUT↓		$^1/_4t_{c(C)} - 10$			$^1/_4t_{c(C)} - 10$			ns
$t_{h(A\text{-}WMD)}$§	Address hold time after $\overline{WE}$↑, $\overline{MEN}$↑, or $\overline{DEN}$↑		0‡			0‡			ns
$t_{su(A\text{-}MD)}$§	Address bus setup time prior to $\overline{MEN}$↓ or $\overline{DEN}$↓		$^1/_4t_{c(C)} - 45$			$^1/_4t_{c(C)} - 35$			ns

‡ This parameter is not production tested.
§ Address bus will be valid upon $\overline{WE}$↑, $\overline{DEN}$↑, or $\overline{MEN}$↑.
NOTE 4: For interfacing I/O devices, see Figure 4.

timing requirements over recommended operating conditions

		TEST CONDITIONS (SEE FIGURE 2)	SMJ320C10-14 SMJ320C10			SMJ320C10-25			UNIT
			MIN	NOM	MAX	MIN	NOM	MAX	
$t_{su(D)}$	Setup time, data bus valid prior to CLKOUT↓	R_L = 825 Ω, C_L = 100 pF	50			40			ns
$t_{h(D)}$	Hold time, data bus held valid after CLKOUT↓ (see Note 5)		0			0			ns

NOTE 5: Data may be removed from the data bus upon $\overline{MEN}$↑ or $\overline{DEN}$↑ preceding CLKOUT↓.

SUGGESTED I/O DECODE CIRCUIT

The circuit shown in Figure 4 is a design example for interfacing I/O devices to the SMJ320C10. This circuit decodes the address for output operations using the OUT instruction. The same circuit can be used to decode input and output operations if the inverter ('ALS04) is replaced with a NAND gate and both $\overline{DEN}$ and $\overline{WE}$ are connected. Inputs and outputs can be decoded at the same port provided the output of the decoder ('AS137) is gated with the appropriate signal ($\overline{DEN}$ or $\overline{WE}$) to select read or write (using an 'ALS32). Access times can be increased when the circuit shown in Figure 4 repeated to support IN instructions with $\overline{DEN}$ connected rather than $\overline{WE}$.

The table write (TBLW) function requires a different circuit. A detailed discussion of an example circuit for this function is described on page 315 of the application report, "Interfacing External Memory to the TMS32010," published in the book, *Digital Signal Processing Applications with the TMS320 Family*. A schematic of this circuit as shown on page 318 of that book.

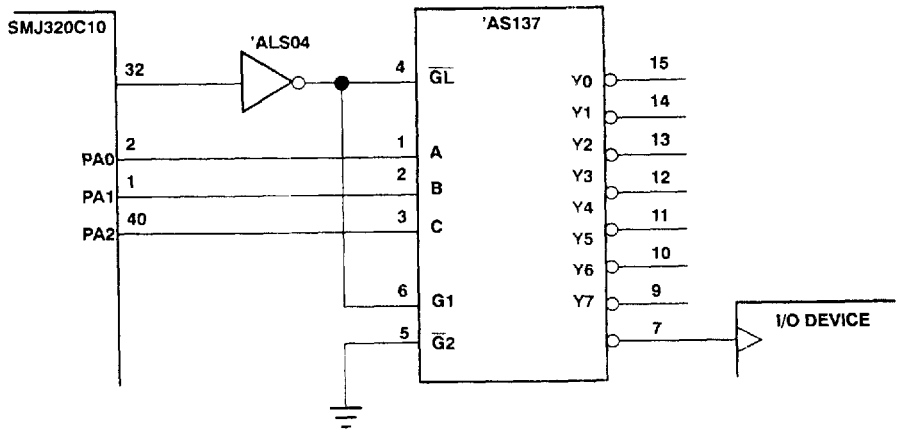


FIGURE 4. I/O DECODE CIRCUIT

RESET ($\overline{RS}$) TIMING

switching characteristics over recommended operating conditions

PARAMETER		TEST CONDITIONS (SEE FIGURE 2)	MIN	TYP	MAX	UNIT
t_{d11}	Delay time, $\overline{DEN}$↑, $\overline{WE}$↑, and $\overline{MEN}$↑ from $\overline{RS}$	R_L = 825 Ω, C_L = 100 pF			$^1/_4t_{c(C)}+50^\dagger$	ns
$t_{dis(R)}$	Data bus disable time after $\overline{RS}$				$^1/_4t_{c(C)}+50^\dagger$	ns

† This parameter is not production tested.

timing requirements over recommended operating conditions

		SMJ320C10-14 SMJ320C10			SMJ320C10-25			UNIT
		MIN	NOM	MAX	MIN	NOM	MAX	
$t_{su(R)}$	Reset ($\overline{RS}$) setup time prior to CLKOUT	50			40			ns
$t_{w(R)}$	$\overline{RS}$ pulse duration	$5t_{c(C)}$			$5t_{c(C)}$			ns

INTERRUPT ($\overline{INT}$) TIMING

timing requirements over recommended operating conditions

		SMJ320C10-14 SMJ320C10			SMJ320C10-25			UNIT
		MIN	NOM	MAX	MIN	NOM	MAX	
$t_{f(INT)}$‡	Fall time, $\overline{INT}$		10			10		ns
$t_{w(INT)}$	Pulse duration, $\overline{INT}$	$t_{c(C)}$			$t_{c(C)}$			ns
$t_{su(INT)}$	Setup time, $\overline{INT}$↓ before CLKOUT↓	50			40			ns

IO ($\overline{BIO}$) TIMING

timing requirements over recommended operating conditions

		SMJ320C10-14 SMJ320C10			SMJ320C10-25			UNIT
		MIN	NOM	MAX	MIN	NOM	MAX	
$t_{f(IO)}$‡	Fall time, $\overline{BIO}$		10			10		ns
$t_{w(IO)}$	Pulse duration, $\overline{BIO}$	$t_{c(C)}$			$t_{c(C)}$			ns
$t_{su(IO)}$	Setup time, $\overline{BIO}$↓ before CLKOUT↓	50			40			ns

‡ $\overline{INT}$ and $\overline{BIO}$ fall times must be less than 15 ns.

electrical characteristics over specified temperature range (unless otherwise noted)

PARAMETER		TEST CONDITIONS	MIN	TYP†	MAX	UNIT
I_{CC}‡ Supply current	SMJ320C15-25	f = 25.6 MHz, V_{CC} = 5.5 V			65	mA
	SMJ320C15	f = 20.5 MHz, V_{CC} = 5.5 V			55	
	SMJ320E15	f = 20.5 MHz, V_{CC} = 5.5 V			90	

† All typical values are at V_{CC} = 5 V, T_A = 25°C.
‡ I_{CC} characteristics are inversely proportional to temperature; i.e., I_{CC} decreases approximately linearly with temperature.

CLOCK CHARACTERISTICS AND TIMING

The SMJ320C15/E15 can use either its internal oscillator or an external frequency source for a clock.

internal clock option

The internal oscillator is enabled by connecting a crystal across X1 and X2/CLKIN (see Figure 1). The frequency of CLKOUT is one-fourth the crystal fundamental frequency. The crystal should be fundamental mode, and parallel resonant, with an effective series resistance of 30 ohms, a power dissipation of 1 mW, and should be specified at a load capacitance of 20 pF.

PARAMETER		TEST CONDITIONS	MIN	NOM	MAX	UNIT
Crystal frequency, f_x¶	SMJ320C15-25	–55°C to 125°C with 8 MHz crystal	6.7		25.6	MHz
	SMJ320C15/E15	–55°C to 125°C with 8 MHz crystal	6.7		20.5	
C1, C2¶		–55°C to 125°C with 8 MHz crystal		10		pF

external clock option

An external frequency source can be used by injecting the frequency directly into X2/CLKIN with X1 left unconnected. The external frequency injected must conform to the specifications listed in the table below.

switching characteristics over recommended operating conditions

PARAMETER		TEST CONDITIONS (SEE FIGURE 2)	SMJ320C15-25			SMJ320C15			SMJ320E15			UNIT
			MIN	NOM	MAX	MIN	NOM	MAX	MIN	NOM	MAX	
$t_{c(C)}$	CLKOUT cycle time§	R_L = 825 Ω, C_L = 100 pF	156.25	160	600	195.12	200	600	195.12	200	600	ns
$t_{r(C)}$	CLKOUT rise time			10			10			10		ns
$t_{f(C)}$	CLKOUT fall time			8			8			8		ns
$t_{w(CL)}$	Pulse duration, CLKOUT low			72			92			92		ns
$t_{w(CH)}$	Pulse duration, CLKOUT high			70			90			90		ns
$t_{d(MCC)}$¶	Delay time, CLKIN↑ to CLKOUT↓		15		40	15		40	20		60	ns

§ $t_{c(C)}$ is the cycle time of CLKOUT, i.e., $4 \times t_{c(MC)}$ (4 times CLKIN cycle time if an external oscillator is used).
¶ This parameter is not production tested.

TEXAS INSTRUMENTS
POST OFFICE BOX 1443 • HOUSTON, TEXAS 77001

timing requirements over recommended operating conditions (unless otherwise noted) (see Note 3)

		SMJ320C15-25			SMJ320C15 SMJ320E15			UNIT
		MIN	NOM	MAX	MIN	NOM	MAX	
$t_{c(MC)}$	Master clock cycle time	39.06	40	150	48.78	50	150	ns
$t_{r(MC)}$	Rise time, master clock input (see Note 3)		5			5		ns
$t_{f(MC)}$	Fall time, master clock input (see Note 3)		5			5		ns
$t_{w(MCP)}$†	Pulse duration, master clock	$0.45t_{c(MC)}$		$0.55t_{c(MC)}$	$0.4t_{c(MC)}$		$0.6t_{c(MC)}$	ns
$t_{w(MCL)}$	Pulse duration, master clock low		15			20		ns
$t_{w(MCH)}$	Pulse duration, master clock high		15			20		ns

† This parameter is not production tested.
NOTE 3: CLKIN rise and fall times must be less than 10 ns.

MEMORY AND PERIPHERAL INTERFACE TIMING

switching characteristics over recommended operating conditions

PARAMETER		TEST CONDITIONS (SEE FIGURE 2)	SMJ320C15-25			SMJ320C15			UNIT
			MIN	NOM	MAX	MIN	NOM	MAX	
t_{d1}	Delay time, CLKOUT↓ to address bus valid	R_L = 825 Ω, C_L = 100 pF	10‡		40	10‡		50	ns
t_{d2}	Delay time, CLKOUT↓ to $\overline{MEN}$↓		$^1/_4t_{c(C)} - 5$‡		$^1/_4t_{c(C)} + 12$	$^1/_4t_{c(C)} - 5$‡		$^1/_4t_{c(C)} + 15$	ns
t_{d3}	Delay time, CLKOUT↓ to $\overline{MEN}$↑		−10‡		12	−10‡		15	ns
t_{d4}	Delay time, CLKOUT↓ to $\overline{DEN}$↓		$^1/_4t_{c(C)} - 5$‡		$^1/_4t_{c(C)} + 12$	$^1/_4t_{c(C)} - 5$‡		$^1/_4t_{c(C)} + 15$	ns
t_{d5}	Delay time, CLKOUT↓ to $\overline{DEN}$↑		−10‡		12	−10‡		15	ns
t_{d6}	Delay time, CLKOUT↓ to $\overline{WE}$↓		$^1/_2t_{c(C)} - 5$‡		$^1/_2t_{c(C)} + 12$	$^1/_2t_{c(C)} - 5$‡		$^1/_2t_{c(C)} + 15$	ns
t_{d7}	Delay time, CLKOUT↓ to $\overline{WE}$↑		−10‡		12	−10‡		15	ns
t_{d8}	Delay time, CLKOUT↓ to data bus OUT valid				$^1/_4t_{c(C)} + 52$			$^1/_4t_{c(C)} + 65$	ns
t_{d9}	Time after CLKOUT↓ that data bus starts to be driven		$^1/_4t_{c(C)} - 5$‡			$^1/_4t_{c(C)} - 5$‡			ns
t_{d10}	Time after CLKOUT↓ that data bus stops being driven				$^1/_4t_{c(C)} + 40$‡			$^1/_4t_{c(C)} + 40$‡	ns
t_v	Data bus OUT valid after CLKOUT↓		$^1/_4t_{c(C)} - 10$			$^1/_4t_{c(C)} - 10$			ns
$t_{h(A\text{-}WMD)}$§	Address hold time after $\overline{WE}$↑, $\overline{MEN}$↑, or $\overline{DEN}$↑		0‡			0‡			ns
$t_{su(A\text{-}MD)}$§	Address bus setup time prior to $\overline{MEN}$↓ or $\overline{DEN}$↓		$^1/_4t_{c(C)} - 35$			$^1/_4t_{c(C)} - 45$			ns

‡ This parameter is not production tested.
§ Address bus will be valid upon $\overline{WE}$↑, $\overline{DEN}$↑, or $\overline{MEN}$↑.
NOTE 4: For interfacing I/O devices, see Figure 4.

switching characteristics over recommended operating conditions (concluded)

PARAMETER		TEST CONDITIONS (SEE FIGURE 2)	SMJ320E15			UNIT
			MIN	NOM	MAX	
t_{d1}	Delay time, CLKOUT↓ to address bus valid	R_L = 825 Ω, C_L = 100 pF	10†		50	ns
t_{d2}	Delay time, CLKOUT↓ to $\overline{MEN}$↓		$^1/_4t_{c(C)}-5$†		$^1/_4t_{c(C)}+15$	ns
t_{d3}	Delay time, CLKOUT↓ to $\overline{MEN}$↑		−10‡		15	ns
t_{d4}	Delay time, CLKOUT↓ to $\overline{DEN}$↓		$^1/_4t_{c(C)}-5$†		$^1/_4t_{c(C)}+15$	ns
t_{d5}	Delay time, CLKOUT↓ to $\overline{DEN}$↑		−10†		15	ns
t_{d6}	Delay time, CLKOUT↓ to $\overline{WE}$↓		$^1/_2t_{c(C)}-5$†		$^1/_2t_{c(C)}+15$	ns
t_{d7}	Delay time, CLKOUT↓ to $\overline{WE}$↑		−10‡		15	ns
t_{d8}	Delay time, CLKOUT↓ to data bus OUT valid				$^1/_4t_{c(C)}+80$	ns
t_{d9}	Time after CLKOUT↓ that data bus starts to be driven		$^1/_4t_{c(C)}-5$†			ns
t_{d10}	Time after CLKOUT↓ that data bus stops being driven				$^1/_4t_{c(C)}+80$†	ns
t_v	Data bus OUT valid after CLKOUT↓		$^1/_4t_{c(C)}-10$			ns
$t_{h(A\text{-}WMD)}$‡	Address hold time after $\overline{WE}$↑, $\overline{MEN}$↑, or $\overline{DEN}$↑		0			ns
$t_{su(A\text{-}MD)}$‡	Address bus setup time prior to $\overline{MEN}$↓ or $\overline{DEN}$↓		$^1/_4t_{c(C)}-45$			ns

† This parameter is not production tested.
‡ Address bus will be valid upon $\overline{WE}$↑, $\overline{DEN}$↑, or $\overline{MEN}$↑
NOTE 4: For interfacing I/O devices, see Figure 4.

timing requirements over recommended operating conditions

		TEST CONDITIONS (SEE FIGURE 2)	SMJ320C15-25			SMJ320C15 SMJ320E15			UNIT
			MIN	NOM	MAX	MIN	NOM	MAX	
$t_{su(D)}$	Setup time, data bus valid prior to CLKOUT↓	R_L = 825 Ω, C_L = 100 pF	40			50			ns
$t_{h(D)}$	Hold time, data bus held valid after CLKOUT↓ (see Note 5)		0			0			ns

NOTE 5: Data may be removed from the data bus upon $\overline{MEN}$↑ or $\overline{DEN}$↑ preceding CLKOUT↓.

TEXAS INSTRUMENTS
POST OFFICE BOX 1443 • HOUSTON, TEXAS 77001

RESET ($\overline{RS}$) TIMING

switching characteristics over recommended operating conditions

PARAMETER		TEST CONDITIONS (SEE FIGURE 2)	MIN	TYP	MAX	UNIT
t_{d11}	Delay time, $\overline{DEN}$↑, $\overline{WE}$↑, and $\overline{MEN}$↑ from $\overline{RS}$	R_L = 825 Ω, C_L = 100 pF			$^1/_2t_{c(C)}+50^\dagger$	ns
$t_{dis(R)}$	Data bus disable time after $\overline{RS}$				$^1/_4t_{c(C)}+50^\dagger$	ns

† This parameter is not production tested.

timing requirements over recommended operating conditions

		SMJ320C15-25			SMJ320C15 SMJ320E15			UNIT
		MIN	NOM	MAX	MIN	NOM	MAX	
$t_{su(R)}$	Reset ($\overline{RS}$) setup time prior to CLKOUT (see Note 6)	40			50			ns
$t_{w(R)}$	$\overline{RS}$ pulse duration	$5t_{c(C)}$			$5t_{c(C)}$			ns

NOTE 6: $\overline{RS}$ can occur anytime during a clock cycle. Time given is minimum to ensure synchronous operation.

INTERRUPT ($\overline{INT}$) TIMING

timing requirements over recommended operating conditions

		SMJ320C15-25			SMJ320C15 SMJ320E15			UNIT
		MIN	NOM	MAX	MIN	NOM	MAX	
$t_{f(INT)}$‡	Fall time, $\overline{INT}$		10			10		ns
$t_{w(INT)}$	Pulse duration, $\overline{INT}$	$t_{c(C)}$			$t_{c(C)}$			ns
$t_{su(INT)}$	Setup time, $\overline{INT}$↓ before CLKOUT↓	40			50			ns

IO ($\overline{BIO}$) TIMING

timing requirements over recommended operating conditions

		SMJ320C15-25			SMJ320C15 SMJ320E15			UNIT
		MIN	NOM	MAX	MIN	NOM	MAX	
$t_{f(IO)}$‡	Fall time, $\overline{BIO}$		10			10		ns
$t_{w(IO)}$	Pulse duration, $\overline{BIO}$	$t_{c(C)}$			$t_{c(C)}$			ns
$t_{su(IO)}$	Setup time, $\overline{BIO}$↓ before CLKOUT↓	40			50			ns

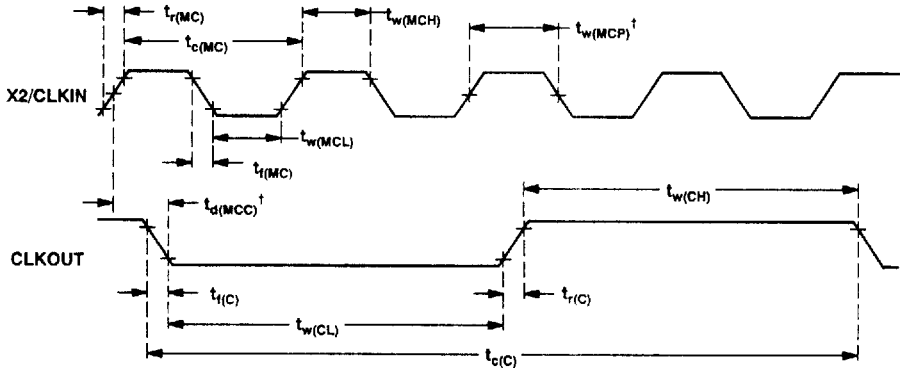
‡ $\overline{INT}$ and $\overline{BIO}$ fall times must be less than 15 ns.

TIMING DIAGRAMS

This section contains all the timing diagrams for the SMJ320 first-generation devices. Refer to the top corner for the specific device.

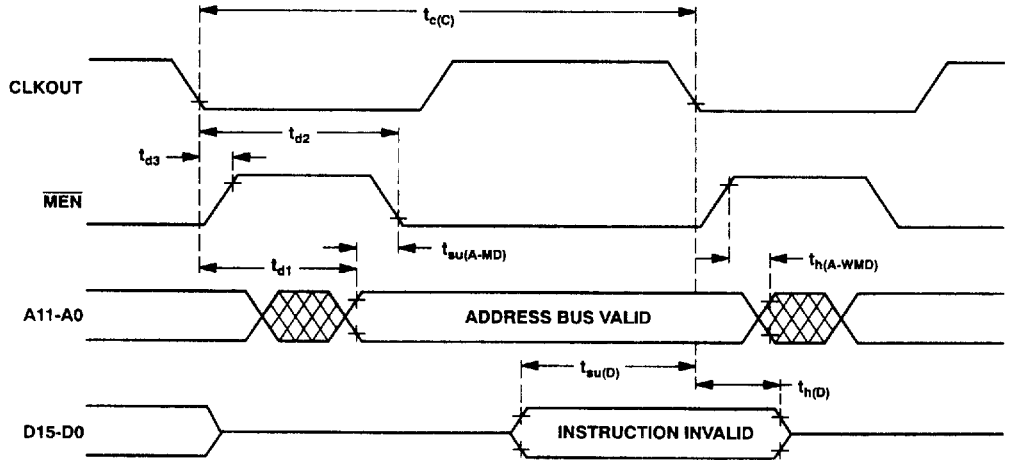
Timing measurements are referenced to and from a low voltage of 0.8 volts and a high voltage of 2.0 volts, unless otherwise noted, See Figure 4.

clock timing



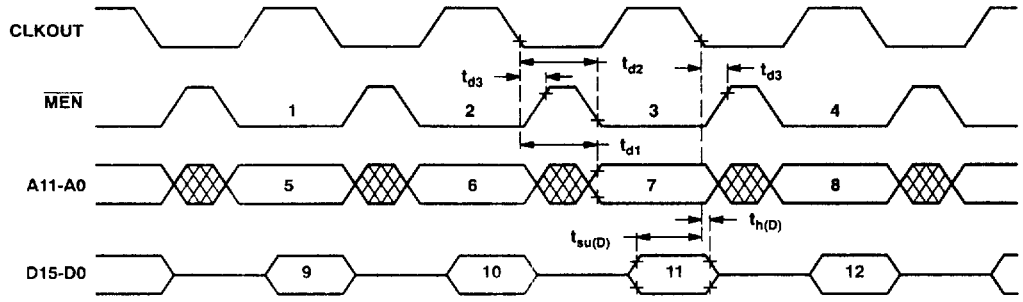
† $t_{d(MCC)}$ and $t_{w(MCP)}$ are referenced to an intermediate level of 1.5-volts on the CLKIN waveform.

memory read timing



TEXAS INSTRUMENTS
POST OFFICE BOX 1443 • HOUSTON, TEXAS 77001

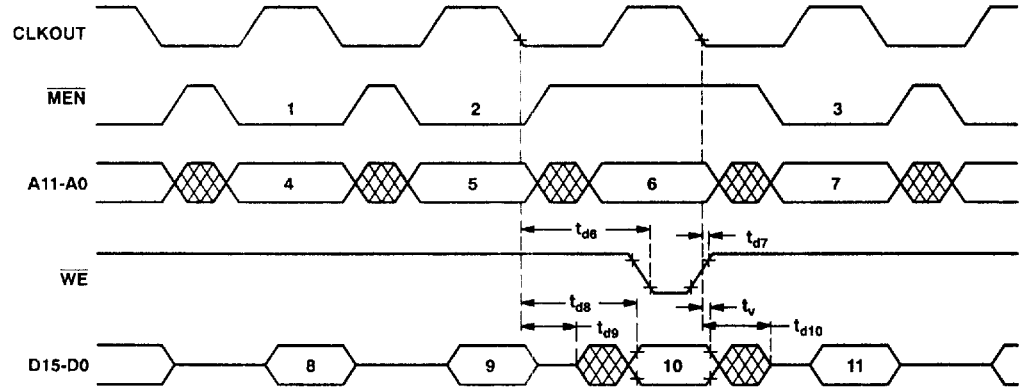
TBLR instruction timing



LEGEND:

1. TBLR INSTRUCTION PREFETCH
2. DUMMY PREFETCH
3. DATA FETCH
4. NEXT INSTRUCTION PREFETCH
5. ADDRESS BUS VALID
6. ADDRESS BUS VALID
7. ADDRESS BUS VALID
8. ADDRESS BUS VALID
9. INSTRUCTION VALID
10. INSTRUCTION VALID
11. DATA INPUT VALID
12. INSTRUCTION INPUT VALID

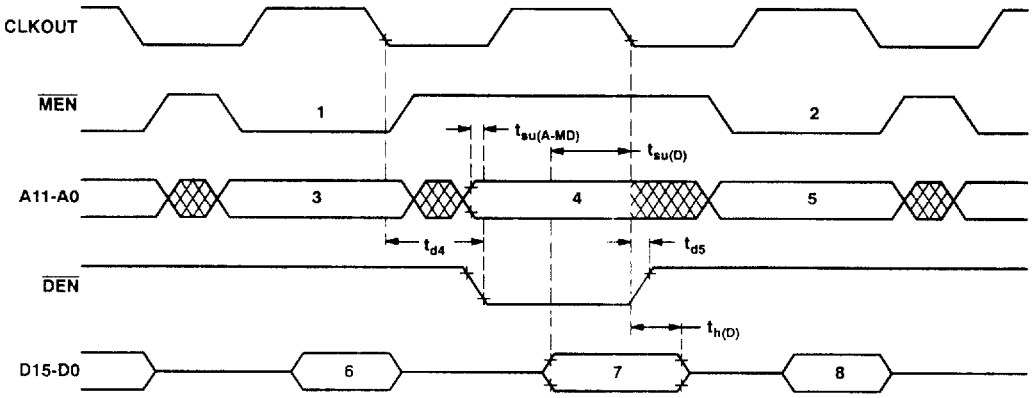
TBLW instruction timing



LEGEND:

1. TBLW INSTRUCTION PREFETCH
2. DUMMY PREFETCH
3. NEXT INSTRUCTION PREFETCH
4. ADDRESS BUS VALID
5. ADDRESS BUS VALID
6. ADDRESS BUS VALID
7. ADDRESS BUS VALID
8. INSTRUCTION VALID
9. INSTRUCTION VALID
10. DATA OUTPUT VALID
11. INSTRUCTION OUTPUT VALID

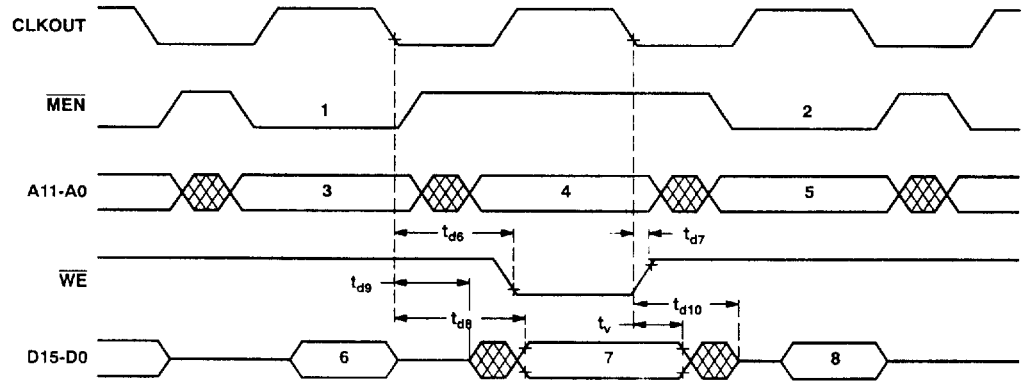
IN instruction timing



LEGEND:

1. IN INSTRUCTION PREFETCH
2. NEXT INSTRUCTION PREFETCH
3. ADDRESS BUS VALID
4. PERIPHERAL ADDRESS VALID
5. ADDRESS BUS VALID
6. INSTRUCTION INPUT VALID
7. DATA INPUT VALID
8. INSTRUCTION INPUT VALID

OUT instruction timing

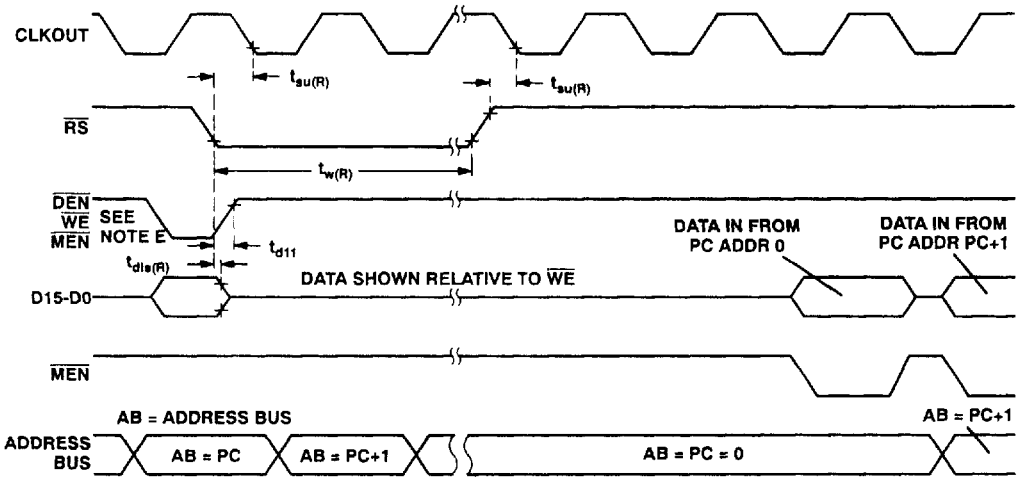


LEGEND:

1. IN INSTRUCTION PREFETCH
2. NEXT INSTRUCTION PREFETCH
3. ADDRESS BUS VALID
4. PERIPHERAL ADDRESS VALID
5. ADDRESS BUS VALID
6. INSTRUCTION INPUT VALID
7. DATA INPUT VALID
8. INSTRUCTION INPUT VALID

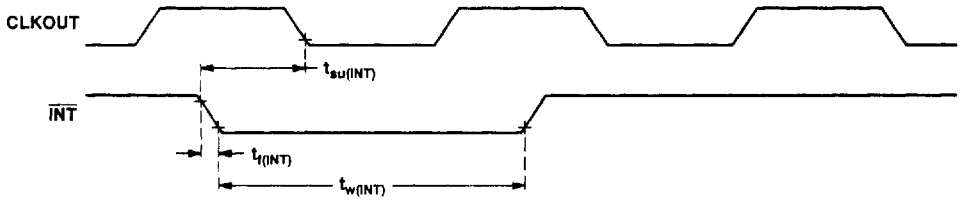
TEXAS INSTRUMENTS
POST OFFICE BOX 1443 • HOUSTON, TEXAS 77001

reset timing

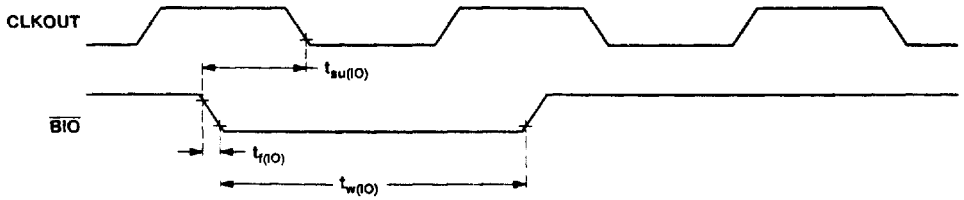


NOTES: A. RS forces DEN, WE, and MEN high and places data bus D0 through D15 in a high-impedance state. AB outputs (and program counter) are synchronously cleared to zero after the next complete CLK cycle from RS↓.
B. RS must be maintained for a minimum of five clock cycles.
C. Resumption of normal program will commence after one complete CLK cycle from RS↑.
D. Due to the synchronization action on RS, time to execute the function can vary dependent upon when RS↑ or RS↓ occur in the CLK cycle.
E. Diagram shown is for definition purpose only. DEN, WE, and MEN are mutually exclusive.
F. During a write cycle, RS may produce an invalid write address.

interrupt timing



BIO timing



EPROM PROGRAMMING

absolute maximum ratings, T_A = 25°C (unless otherwise noted)†

Supply voltage range, V_{PP} (see Note 1) ..... −0.6 V to 14 V

† Stresses beyond those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. This is a stress rating only, and functional operation of the device at these or any other conditions beyond those indicated in the "Recommended Operating Conditions" section of this specification is not implied. Exposure to absolute-maximum-rated conditions for extended periods may affect device reliability.

NOTE 1: Under "Absolute Maximum Ratings", all voltage values are with respect to V_{SS}.

recommended operating conditions

		MIN	NOM	MAX	UNIT
V_{PP}	Supply voltage (see Note 7)	12.25	12.5	12.75	V

NOTE 7: V_{PP} can be connected to V_{CC} directly (except in the program mode) V_{CC} supply current in this case would be $I_{CC} + I_{PP}$. During programming, V_{PP} must be maintained at 12.5 V (±0.25 V).

electrical characteristics over specified temperature range (unless otherwise noted)

PARAMETER		TEST CONDITIONS	MIN	TYP	MAX	UNIT
I_{PP1}	V_{PP} supply current	$V_{PP} = V_{CC} = 5.25$ V			100	µA
I_{PP2}	V_{PP} supply current (during program pulse)	$V_{PP} = 12.75$ V			50	mA

recommended timing requirements for programming, T_A = 25°C, V_{CC} = 6 V, V_{PP} = 12.5 V, (see Note 8)

		MIN	NOM	MAX	UNIT
$t_{w(IPGM)}$	Initial program pulse duration	0.95	1	1.05	ms
$t_{w(FPGM)}$	Final pulse duration	2.85		78.75	ms
$t_{su(A)}$	Address setup time		10		µs
$t_{su(E)}$	$\overline{E}$ setup time		10		µs
$t_{su(G)}$	$\overline{G}$ setup time		1.5		µs
$t_{dis(G)}$	Output disable time from $\overline{G}$ (see Note 9)		90		ns
$t_{en(G)}$	Output enable time from $\overline{G}$		105		ns
$t_{su(D)}$	Data setup time		4		µs
$t_{su(VPP)}$	V_{PP} setup time		15		µs
$t_{su(VCC)}$	V_{CC} setup time		20		µs
$t_{h(A)}$	Address hold time		8		µs
$t_{h(D)}$	Data hold time		2.8		µs

NOTES: 8. For all switching characteristics and timing measurements, input pulse levels are 0.40 V to 2.4 V and V_{PP} = 12.5 V ± 0.25 V during programming.

9. Common test conditions apply for $t_{dis(G)}$ except during programming.

TEXAS INSTRUMENTS
POST OFFICE BOX 1443 • HOUSTON, TEXAS 77001

PROGRAMMING THE SMJ320E15 EPROM CELL

The SMJ320E15 includes a 4K x 16-bit industy-standard EPROM cell for prototyping, early field testing, and low-volume production. The SMJ320C15 with a 4K-word masked ROM then provides a migration path for cost-effective production. An EPROM programmer adaptor socket (part #RTC/PGM320A-06), shown in Figure 5, is available to provide 40-pin to 28-pin conversion for programming the SMJ320E15.

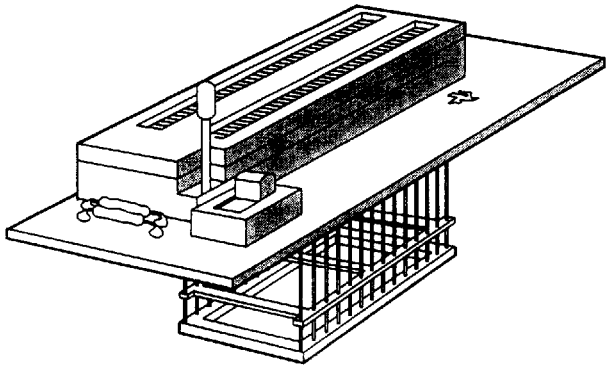


FIGURE 5. EPROM PROGRAMMER ADAPTOR SOCKET

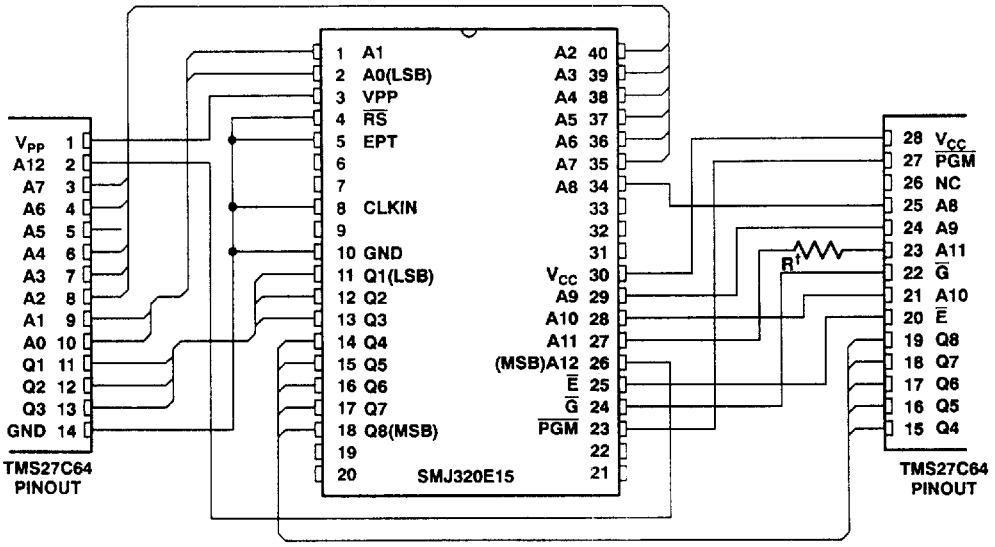
Key features of the EPROM cell include the normal programming operation as well as verification. The EPROM cell also includes a code protection feature that allows code to be protected against copyright violations.

The SMJ320E15 EPROM cell is programmed using the same family and device pinout codes as the TMS27C64 8K x 8-bit EPROM. The TMS27C64 EPROM series are unltraviolet-light erasable, electrically programmable, read-only memories, fabricated using HVCMOS technology. They are pin-compatible with existing 28-pin ROMs and EPROMs. These EPROMs operate from a single 5-V supply in the read mode; however, 12.5-V V_{PP} supplies are needed for programming. All programming signals are TTL level. For programming outside the system, existing EPROM programmers can be used. Locations may be programmed singly, in blocks, or at random.

Figure 6 shows the wiring conversion to program the SMJ320E15 using the 28-pin pinout of the TMS27C64. Table 5 on pin nomenclature provides a description of the TMS27C64 pins. The code to be programmed into the device should be in serial mode. The SMJ320E15 uses 13 address lines to address 4K-word memory in byte format.

TABLE 5. PIN NOMENCLATURE (SMJ320E15)

NAME	I/O	DEFINITION
A12(MSB)-A0(LSB)	I	On-chip EPROM programming address lines
CLKIN	I	Clock oscillator input
$\overline{E}$	I	EPROM chip select
EPT	I	EPROM test mode select
$\overline{G}$	I	EPROM read/verify select
GND	I	Ground
$\overline{PGM}$	I	EPROM write/program select
Q8(MSB)-Q1(LSB)	I/O	Data lines for byte-wide programming of on-chip 8K bytes of EPROM
$\overline{RS}$	I	Reset for initializing the device
V_{CC}	I	5-V power supply
V_{PP}	I	12.5-V power supply



† R = 3.9k ohms.

FIGURE 6. SMJ320E15 EPROM PROGRAMMING CONVERSION TO TMS27C64 EPROM PINOUT

TEXAS INSTRUMENTS
POST OFFICE BOX 1443 • HOUSTON, TEXAS 77001

Table 6 shows the programming levels required for programming, verifying, reading, and protecting the EPROM cell.

TABLE 6. SMJ320E15 PROGRAMMING MODE LEVELS

SIGNAL NAME	SMJ320E15 PIN	TMS27C64 PIN	PROGRAM	VERIFY	READ	PROTECT VERIFY	ROM PROTECT
$\overline{E}$	25	20	V_{IL}	V_{IL}	V_{IL}	V_{IL}	V_{IH}
$\overline{G}$	24	22	V_{IH}	$\overline{PULSE}$	$\overline{PULSE}$	V_{IL}	V_{IH}
$\overline{PGM}$	23	27	$\overline{PULSE}$	V_{IH}	V_{IH}	V_{IH}	V_{IH}
V_{PP}	3	1	V_{PP}	V_{PP}	V_{CC}	V_{CC} + 1	V_{PP}
V_{CC}	30	28	V_{CC}	V_{CC}	V_{CC}	V_{CC} + 1	V_{CC} + 1
V_{SS}	10	14	V_{SS}	V_{SS}	V_{SS}	V_{SS}	V_{SS}
CLKIN	8	14	V_{SS}	V_{SS}	V_{SS}	V_{SS}	V_{SS}
$\overline{RS}$	4	14	V_{SS}	V_{SS}	V_{SS}	V_{SS}	V_{SS}
EPT	5	26	V_{SS}	V_{SS}	V_{SS}	V_{PP}	V_{PP}
Q1-Q8	18-11	19-15, 13-11	D_{IN}	Q_{OUT}	Q_{OUT}	Q8=RBIT	Q8=$\overline{PULSE}$
A12-A10	26, 27, 28	2, 23, 21	ADDR	ADDR	ADDR	X	X
A9-A7	29, 34, 35	24, 25, 3	ADDR	ADDR	ADDR	X	X
A6	36	4	ADDR	ADDR	ADDR	V_{IL}	X
A5	37	5	ADDR	ADDR	ADDR	X	X
A4	38	6	ADDR	ADDR	ADDR	X	V_{IH}
A3-A0	39, 40, 1, 2	7-10	ADDR	ADDR	ADDR	X	X

LEGEND:

V_{IH} = TTL high level; V_{IL} = TTL low level; ADDR = byte address bit
V_{PP} = 12.5 V ± 0.25 V; V_{CC} = 5 V ± 5% for read only, otherwise, V_{CC} = 6 V; X = don't care
$\overline{PULSE}$ = low-going TTL level pulse; D_{IN} = byte to be programmed at ADDR
Q_{OUT} = byte stored at ADDR; RBIT = ROM protect bit

programming

Since every memory bit in the cell is a logic 1, the programming operation reprograms certain bits to 0. Once programmed, these bits can only be erased using ultraviolet light. The correct byte is placed on the data bus with V_{PP} set to the 12.5-V level. The $\overline{PGM}$ pin is then pulsed low to program in the zeroes.

erasure

Before programming, the device must be erased by exposing it to ultraviolet light. The recommended minimum exposure dose (UV-intensity × exposure-time) is 15 watt-seconds per square centimeter. A typical 12 milliwatt-seconds per square centimeter, filterless UV lamp will erase the device in 21 minutes. The lamp should be located about 2.5 centimeters above the chip during erasure. After exposure, all bits are in the high state.

verify/read

To verify correct programming, the EPROM cell can be read using either the verify or read line definitions shown in Table 6, assuming the inhibit bit has not been programmed.

program inhibit

Programming may be inhibited by maintaining a high level input on the $\overline{E}$ pin or $\overline{PGM}$ pin.

read

The EPROM contents may be read independent of the programming cycle, provided the RBIT (ROM protect bit) has not been programmed. The read is accomplished by setting $\overline{E}$ to zero and pulsing $\overline{G}$ low. The contents of the EPROM location selected by the value on the address inputs appear on Q8-Q1.

output disable

During the EPROM programming process, the EPROM data outputs may be disabled, if desired, by establishing the output disable state. This state is selected by setting $\overline{G}$ and $\overline{E}$ pins high. While output disable is selected, Q8-Q1are placed in the high-impedance state.

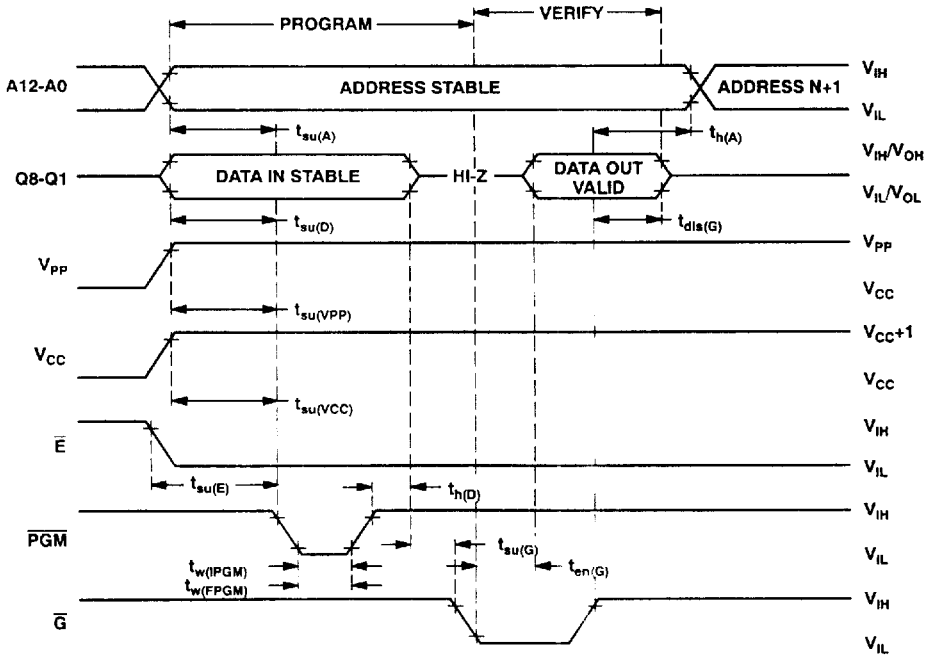
EPROM protection

To protect the proprietary algorithms existing in the code programmed on-chip, the ability to read or verify code from external accesses can be completely disabled. Programming the RBIT disables external access of the EPROM cell and disables the microprocessor mode, making it impossible to access the code resident in the EPROM cell. The only way to remove this protection is to erase the entire EPROM cell, thus removing the proprietary information. The signal requirements for programming this bit are shown in Table 6. The cell can be determined as protected by verifying the programming of the RBIT shown in the table.

standard programming procedure

Before programming, the device must first be completely erased. Then the device can be programmed with the correct code. It is advisable to program unused sections with zeroes as a further security measure. After the programming is complete, the code programmed into the cell should be verified. If the cell passes verification, the next step is to program the ROM protect bit (RBIT). Once the RBIT programming is verified, an opaque label should be placed over the window to protect the EPROM cell from inadvertent erasure by ambient light. At this point, the programming is complete, and the device is ready to be placed into its destination circuit.

program cycle timing



TEXAS INSTRUMENTS
POST OFFICE BOX 1443 • HOUSTON, TEXAS 77001

PACKAGE TYPES

PACKAGE TYPE	SUFFIX	FAMILY MEMBERS
40-pin side-brazed ceramic DIP	JD	SMJ320C10, SMJ320C10-14, SMJ320C10-25 SMJ320C15, SMJ320C15-25
40-pin windowed ceramic DIP	JD	SMJ320E15
44-pad leadless ceramic chip carrier	FD	SMJ320C10, SMJ320C10-14, SMJ320C10-25 SMJ320C15, SMJ320C15-25

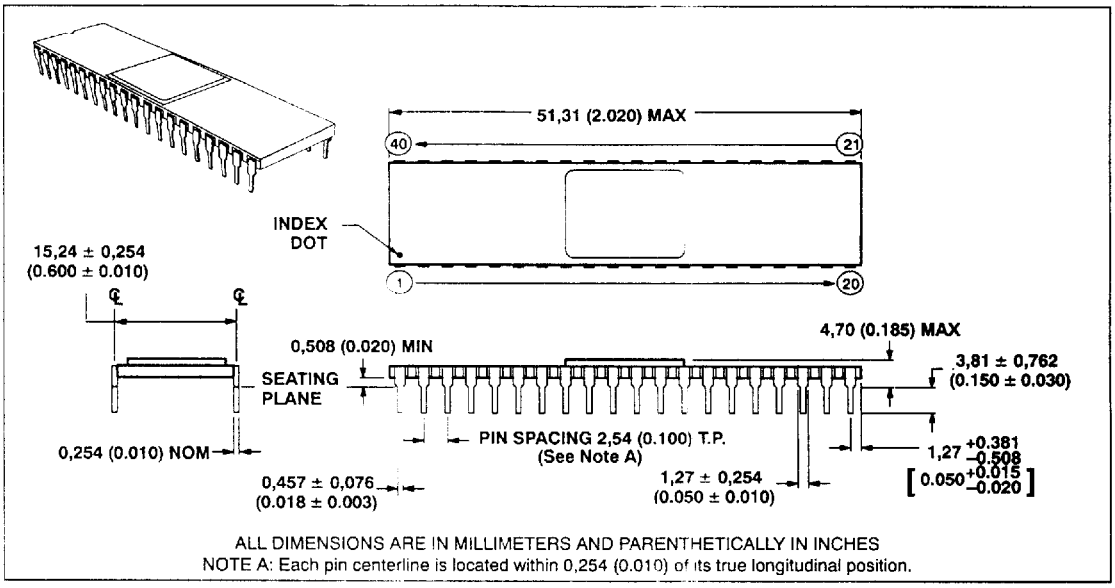
THERMAL DATA

thermal resistance characteristics

PACKAGE	$R_{\theta JA}$ (°C/W)	$R_{\theta JC}$ (°C/W)
40-pin side-brazed ceramic DIP	42.8	7.2
40-pin windowed DIP	41.0	8.0
44-pad leadless ceramic chip carrier	44.7	13.3

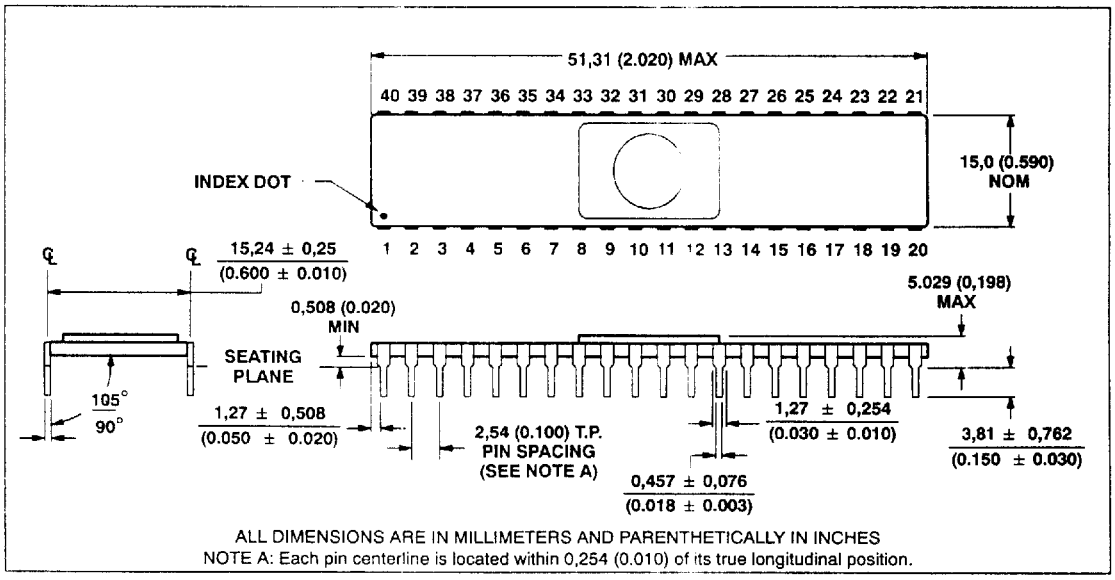
MECHANICAL DATA

40-pin JD ceramic dual-in-line package (SMJ320C10/C15)



ALL DIMENSIONS ARE IN MILLIMETERS AND PARENTHETICALLY IN INCHES
NOTE A: Each pin centerline is located within 0,254 (0.010) of its true longitudinal position.

40-pin JD windowed ceramic dual-in-line package (SMJ320E15)

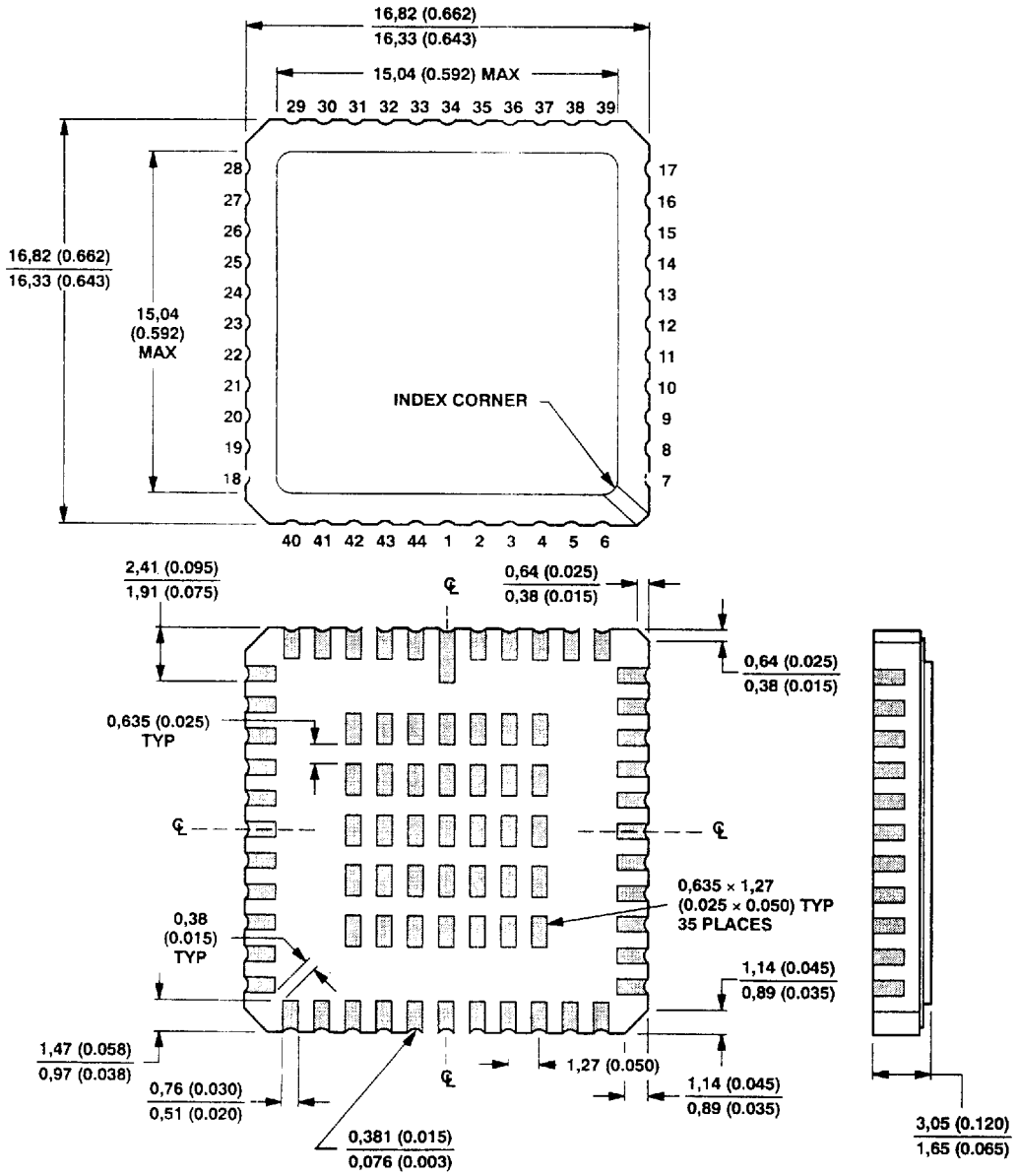


ALL DIMENSIONS ARE IN MILLIMETERS AND PARENTHETICALLY IN INCHES
NOTE A: Each pin centerline is located within 0,254 (0.010) of its true longitudinal position.

TEXAS INSTRUMENTS
POST OFFICE BOX 1443 • HOUSTON, TEXAS 77001

MECHANICAL DATA

44-pad ceramic chip carrier package



The checkerboard pattern is aligned vertically and is symmetrical horizontally as shown.
ALL DIMENSIONS ARE IN MILLIMETERS AND PARENTHETICALLY IN INCHES

TEXAS INSTRUMENTS
POST OFFICE BOX 1443 • HOUSTON, TEXAS 77001

Appendix C

ROM Codes

Note:

Throughout this book, 'C14 designates all devices with a 14 suffix (C14/E14/P14), 'C15 all devices with a 15 suffix (C15/E15/LC15/P15), and 'C17 all devices with a 17 suffix (C17/E17/LC17/P17), unless otherwise noted.

If used often, the routine or entire algorithm can be programmed into the on-chip ROM of a TMS320C1x processor. TMS320 programs can also be expanded by using external memory; this reduces chip count and allows for a more flexible program memory. Multiple functions are easily implemented by a single device, thus enhancing system capabilities.

TMS320 development tools are used to develop, test, refine, and finalize the algorithms. The microcomputer/microprocessor (MC/$\overline{\text{MP}}$) mode is available on all ROM-coded TMS320 devices when accessing either on-chip or off-chip memory is required. The microprocessor mode is used to develop, test, and refine a system application. In this mode of operation, the TMS320 acts as a standard microprocessor by using external program memory. When the algorithm has been finalized, the designer may submit the code to Texas Instruments for masking into the on-chip program ROM. At that time, the TMS320 becomes a microcomputer that executes customized programs out of the on-chip ROM. Should the code need changing or upgrading, the TMS320 may once again be used in the microprocessor mode. This shortens the field upgrade time and avoids the possibility of inventory obsolescence.

A TMS320E1x signal processor with the EPROM option is the solution for low-volume production orders. The EPROM option allows for form-factor emulation. Field upgrades and changes are possible with the EPROM option.

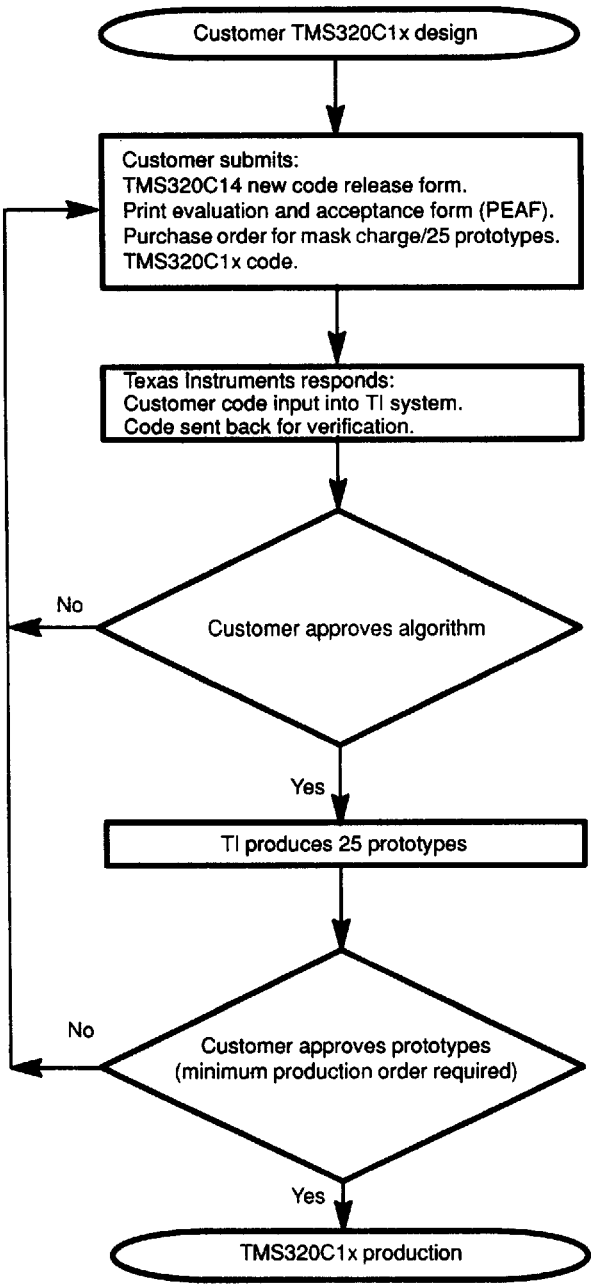
Size of a printed circuit board must be considered in many DSP applications. To use board space efficiently, Texas Instruments offers two options that reduce the chip count providing a single-chip solution. These options incorporate 4K words of on-chip program from either a mask programmable ROM or an EPROM. This allows you to use a code-customized processor for a specific application while taking advantage of the following:

- Greater memory expansion
- Lower system cost
- Less hardware and wiring
- Smaller PCB

Note:

In all ROM-coded devices, a 96-word block of memory located at the highest address space is reserved for device testing. If the program code requires a full 4K × 16-bit memory, external memory or EPROM version (TMS320E1x) must be used.

Figure C–1. TMS320C1x ROM Code Flowchart



Lead times for the first 25 prototype units begin when the customer has formally verified that TI has recorded his code correctly. Leadtimes for the first production order begin once the customer formally approves the masked prototypes. The typical lead time for masked TMS320C1x prototypes is 8 weeks and for masked TMS320C1x production 10 to 12 weeks. Texas Instruments constantly strives to improve these lead times and reserves the right to make changes at any time. Please contact the nearest TI Sales Office for current lead times, mask-ROM production charges, minimum order, and confirmation of the mask/production requirements.

Figure C–1 illustrates the procedure flow for implementing TMS320C14 masked parts.

TMS320C1x ROM code may be submitted in one of the following formats (the preferred media is 5 1/4-in floppy diskettes):

- PROM: TBP28S166, TBP28S86
- EPROM: TMS27C64, TMS2508, TMS2516, TMS2532, TMS2564
- Floppy diskette: TI cross-assembler format
- Modem (BBS): TI cross-assembler format

When a code is submitted to Texas Instruments for a masked device, the code is reformatted to accommodate the TI mask generation system. System level verification by the customer is therefore necessary. Although the code has been reformatted, it is important that the changes remain transparent to you and not affect the execution of the algorithm. The formatting changes made involve deletion of all address tags (unnecessary in a ROM code device) and addition of data in the reserved locations of the ROM for device ROM test. Note that because these changes have been made, a checksum comparison is not a valid means of verification.

With each masked device order, you must sign a disclaimer stating:

"The units to be shipped against this order were assembled, for expediency purposes, on a prototype (that is, non-production qualified) manufacturing line, the reliability of which is not fully characterized. Therefore, the anticipated inherent reliability of these prototype units cannot be expressly defined."

and a release stating:

"Any masked ROM device may be resymbolized as TI standard product and resold as though it were an unprogrammed version of the device, at the convenience of Texas Instruments."

ROM codes will be deleted from the TI system after one year from the last delivery.

Appendix D

Quality and Reliability

Note:

Throughout this book, 'C14 designates all devices with a 14 suffix (C14/E14/P14), 'C15 all devices with a 15 suffix (C15/E15/LC15/P15), and 'C17 all devices with a 17 suffix (C17/E17/LC17/P17), unless otherwise noted.

The quality and reliability performance of Texas Instruments Microprocessor and Microcontroller Products, which includes the three generations of TMS320 digital signal processors, relies on feedback from:

- Our customers
- Our total manufacturing operation from front-end wafer fabrication to final shipping inspection
- Product quality and reliability monitoring.

Our customer's perception of quality must be the governing criterion for judging performance. This concept is the basis for Texas Instruments Corporate Quality Policy, which is as follows:

"For every product or service we offer, we shall define the requirements that solve the customer's problems, and we shall conform to those requirements without exception."

Texas Instruments offers a leadership reliability qualification system, based on years of experience with leading-edge memory technology as well as years of research into customer requirements. Quality and reliability programs at TI are therefore based on customer input and internal information to achieve constant improvement in quality and reliability.

Note:

Without notice, Texas Instruments reserves the right to make changes in MOS Semiconductor test expectations, procedures, and processing. Unless prior arrangements for notification have been made, TI advises all customers to verify current testing and manufacturing conditions prior to relying upon any published data.

D.1 Reliability Stress Tests

Accelerated stress tests are performed on new semiconductor products and process changes to ensure product reliability excellence. The typical test environments used to qualify new products or major changes in processing are:

- High-temperature operating life
- Storage life
- Temperature cycling
- Biased humidity
- Autoclave
- Electrostatic discharge
- Package integrity
- Electromigration
- Channel-hot electrons (performed on geometries less than 2.0 μm).

Typical events or changes that require internal requalification of product include:

- New die design, shrink, or layout
- Wafer process (baseline/control systems, flow, mask, chemicals, gases, dopants, passivation, or metal systems)
- Packaging assembly (baseline control systems or critical assembly equipment)
- Piece parts (such as lead frame, mold compound, mount material, bond wire, or lead finish)
- Manufacturing site.

TI reliability control systems extend beyond qualification. Total reliability controls and management include a product reliability monitor and final product release controls. MOS memories, utilizing high-density active elements, serve as leading indicators in wafer-process integrity at TI MOS fabrication sites, enhancing all MOS logic device yields and reliability. Thousands of MOS devices per month are randomly tested to ensure product reliability and excellence.

Table D–1 lists the microprocessor and microcontroller reliability tests, the duration of the test, and sample size. The following defines and describes those tests in the table.

AOQ (Average outgoing quality)	Number of defective products in a population, usually expressed in terms of parts per million (PPM).
FIT (Failure in time)	Estimated field failure rate in number of failures per billion power-on device hours; 1000 FITS equals 0.1 percent fail per 1000 device hours.
Operating lifetest	Device dynamically exercised at a high ambient temperature (usually 125°C) to simulate field usage that would expose the device to a much lower ambient temperature (such as 55°C). Using a derived high temperature, a 55°C ambient failure rate can be calculated.
High-temperature storage	Device exposed to 150°C unbiased condition. Bond integrity is stressed in this environment.
Biased humidity	Moisture and bias used to accelerate corrosion-type failures in plastic packages. Conditions include 85°C ambient temperature with 85-percent relative humidity (RH). Typical bias voltage is +5 V and ground on alternating pins.
Autoclave (pressure cooker)	Plastic-packaged devices exposed to moisture at 121°C using a pressure of one atmosphere above normal pressure. The pressure forces moisture permeation of the package and accelerates corrosion mechanisms (if present) on the device. External package contaminates can also be activated and caused to generate inter-pin current leakage paths.
Temperature cycle	Device exposed to severe temperature extremes in an alternating fashion (–65°C for 15 minutes and 150°C for 15 minutes per cycle) for at least 1000 cycles. Package strength, bond quality, and consistency of assembly process are stressed in this environment.

Thermal shock	Test similar to the temperature cycle test, but involving a liquid-to-liquid transfer, per MIL-STD-883C, Method 1011.
PIND	Particle Impact Noise Detection test. A non-destructive test to detect loose particles inside a device cavity.
Mechanical Sequence:	
Fine and gross leak	Per MIL-STD-883C, Method 1014.5
Mechanical shock	Per MIL-STD-883C, Method 2002.3, 1500g, 0.5 ms, Condition B
PIND (optional)	Per MIL-STD-883C, Method 2020.4
Vibration, variable frequency	Per MIL-STD-883C, Method 2007.1, 20 g, Condition A
Constant acceleration	Per MIL-STD-883C, Method 2001.2, 20 kg, Condition D, Y1 Plane min
Fine and gross leak	Per MIL-STD-883C, Method 1014.5
Electrical test	To data sheet limits
Thermal Sequence:	
Fine and gross leak	Per MIL-STD-883C, Method 1014.5
Solder heat (optional)	Per MIL-STD-750C, Method 1014.5
Temperature cycle	Per MIL-STD-883C, Method 1010.5, (10 cycles minimum) –65 to +150°C, Condition C
Thermal shock	Per MIL-STD-883C, Method 1011.4,(10 cycles minimum) –55 to +125°C, Condition B
Moisture resistance	Per MIL-STD-883C, Method 1004.4
Fine and gross leak	Per MIL-STD-883C, Method 1014.5
Electrical test	To data sheet limits
Thermal/Mechanical Sequence:	
Fine and gross leak	Per MIL-STD-883C, Method 1014.5
Temperature cycle	Per MIL-STD-883C, Method 1010.5, (10 cycles minimum) –65 to +150°C, Condition C
Constant acceleration	Per MIL-STD-883C, Method 2001.2, 30 kg, Y1 Plane
Fine and gross leak	Per MIL-STD-883C, Method 1014.5
Electrical test	To data sheet limits
Electrostatic discharge	Per MIL-STD-883C, Method 3015

Solderability	Per MIL-STD-883C, Method 2003.3
Solder heat	Per MIL-STD-750C, Method 2031,10 sec
Salt atmosphere	Per MIL-STD-883C, Method 1009.4, Condition A, 24 hrs min
Lead pull	Per MIL-STD-883C, Method 2004.4, Condition A
Lead integrity	Per MIL-STD-883C, Method 2004.4, Condition B1
Electromigration	Accelerated stress testing of conductor patterns to ensure acceptable lifetime of power-on operation
Resistance to solvents	Per MIL-STD-883C, Method 2015.4

Table D–1. Microprocessor and Microcontroller Tests

Test	Duration	Sample Size	
		Plastic	Ceramic
Operating life, 125°C, 5.0 V	1000 hrs	129	129
Operating life, 150°C 5.0 V	1000 hrs	77*	77
Storage life, 150°C	1000 hrs	77	77
Biased 85°C/85 percent RH, 5.0 V	1000 hrs	129	–
Autoclave, 121°C, 1 ATM	240 hrs	77	–
Temperature cycle, –65°C to 150°C	1000 cycles	129	129
Temperature cycle, 0°C to 125°C	3000 cycles	129	129
Thermal shock, –65°C to 150°C	200 cycles	129	129
Electrostatic discharge, ±2 kV		12	12
Latch-up (CMOS devices only)		5	5
Mechanical sequence		–	38
Thermal sequence		–	38
Thermal/mechanical sequence		–	38
PIND		–	45

Table D–1. Microprocessor and Microcontroller Tests (Continued)

Test	Duration	Sample Size	
		Plastic	Ceramic
Internal water vapor		–	3
Solderability		22	22
Solder heat		22	22
Resistance to solvents		15	15
Lead integrity		15	15
Lead pull		22	–
Lead finish adhesion		15	15
Salt atmosphere		15	15
Flammability (UL94-V0)		3	–
Thermal impedance		5	5

Note: If junction temperature does not exceed plasticity of package.

Table D–2 provides a list of the TMS320C1x devices, the approximate number of transistors, and its equivalent number of gates. The numbers were determined during the stages of design verification and production.

Table D–2. TMS320C1x Transistors

Device	# Transistors	# Gates
NMOS:		
TMS32010 (all speeds)	50K	17K
CMOS:		
TMS320C10 (all speeds)	58K	15K
TMS320C14 (all speeds)	122K	25K
TMS320E14 (all speeds)	125K	26K
TMS320C15 (all speeds)	110K	20K
TMS320E15 (all speeds)	113K	21K
TMS320C16 (all speeds)	125K	26K
TMS320C17 (all speeds	115K	22K
TMS320E17 (all speeds)	118K	23K

Appendix E

Development Support

Note:

Throughout this book, 'C14 designates all devices with a 14 suffix (C14/E14/P14), 'C15 all devices with a 15 suffix (C15/E15/LC15/P15), and 'C17 all devices with a 17 suffix (C17/E17/LC17/P17), unless otherwise noted.

Texas Instruments offers an extensive line of development tools for the TMS320C1x generation of DSPs, including tools to evaluate the performance of the processors, generate code, develop algorithm implementations, and fully integrate and debug software and hardware modules.

The following products support development of TMS320C1x-based applications:

Code Generation Tools:

Assembler/linker

System Integration and Debug Tools:

Software simulator
Evaluation module
In-circuit emulator (XDS/22)
Analog interface board

Each TMS320C1x support product is described in the *TMS320 Family Development Support Reference Guide* (literature number SPRU011B). In addition, more than 100 TMS320 third-party developers provide support products to complement TI's offering. For more information on third-party support refer to the *TMS320 Third Party Reference Guide* (literature number SPRU052). To request a copy of either document, contact the TI Customer Response Center at (800) 336-5236.

For information on pricing and availability, contact the nearest TI Field Sales Office or authorized distributor.

E.1 Device and Development Support Tool Nomenclature

To designate the stages in the product development cycle, Texas Instruments assigns prefixes to the part numbers of all TMS320 devices and support tools. Each TMS320 member has one of three prefix designators: TMX, TMP, and TMS. Texas Instruments recommends two of three possible prefix designators for its support tools: TMDX and TMDS. These prefixes represent one of the evolutionary stages of product development from engineering prototypes (TMX/TMDX) through fully qualified production devices/tools (TMS/TMDS). This development flow is defined below.

TMX Experimental device that is not necessarily representative of the final device's electrical specifications.

TMP Final silicon die that conforms to the device's electrical specifications but has not completed quality and reliability verification.

TMS Fully qualified production device.

Support Tool Development Evolutionary Flow:

TMDX Development support product that has not yet completed Texas Instruments internal qualification testing.

TMDS Fully qualified development support product.

TMX and TMP devices and TMDX development support tools are shipped with the following disclaimer:

"Developmental product is intended for internal evaluation purposes."

TMS devices and TMDS development support tools have been fully characterized, and the quality and reliability of the devices have been fully demonstrated. Texas Instruments standard warranty applies.

Note:

Predictions show that prototype devices (TMX or TMP) will have a greater failure rate than the standard production devices. Texas Instruments recommends that these devices *not* be used in any production system because their expected end-use failure rate is still undefined. Only qualified production devices are to be used.

TI device nomenclature also includes a suffix with the device family name. This suffix indicates the package type (for example, N, FN, or GB) and temperature range (for example, L). Figure E–1 provides a legend for reading the complete device name for any TMS320 family device.

Figure E–1. Device Nomenclature

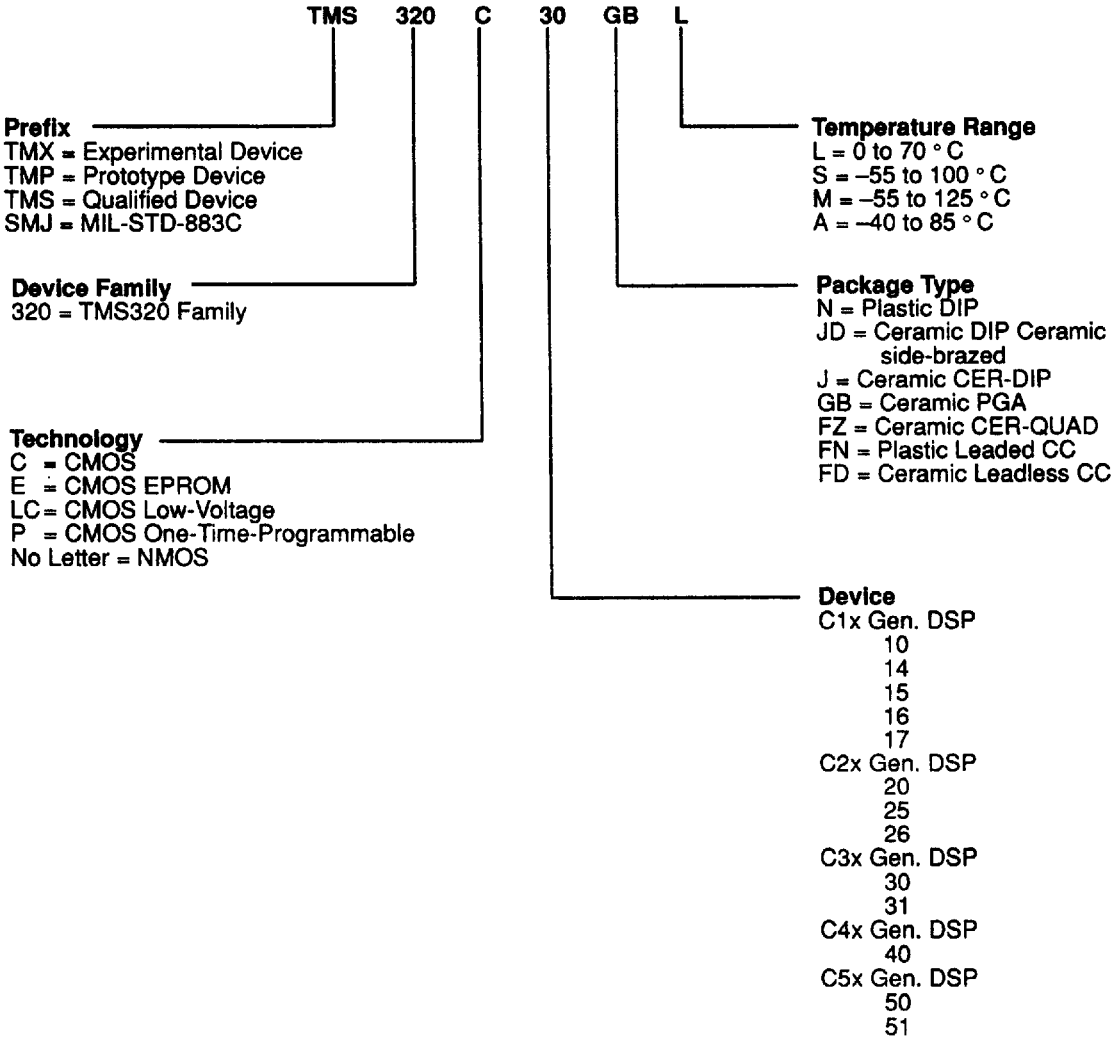
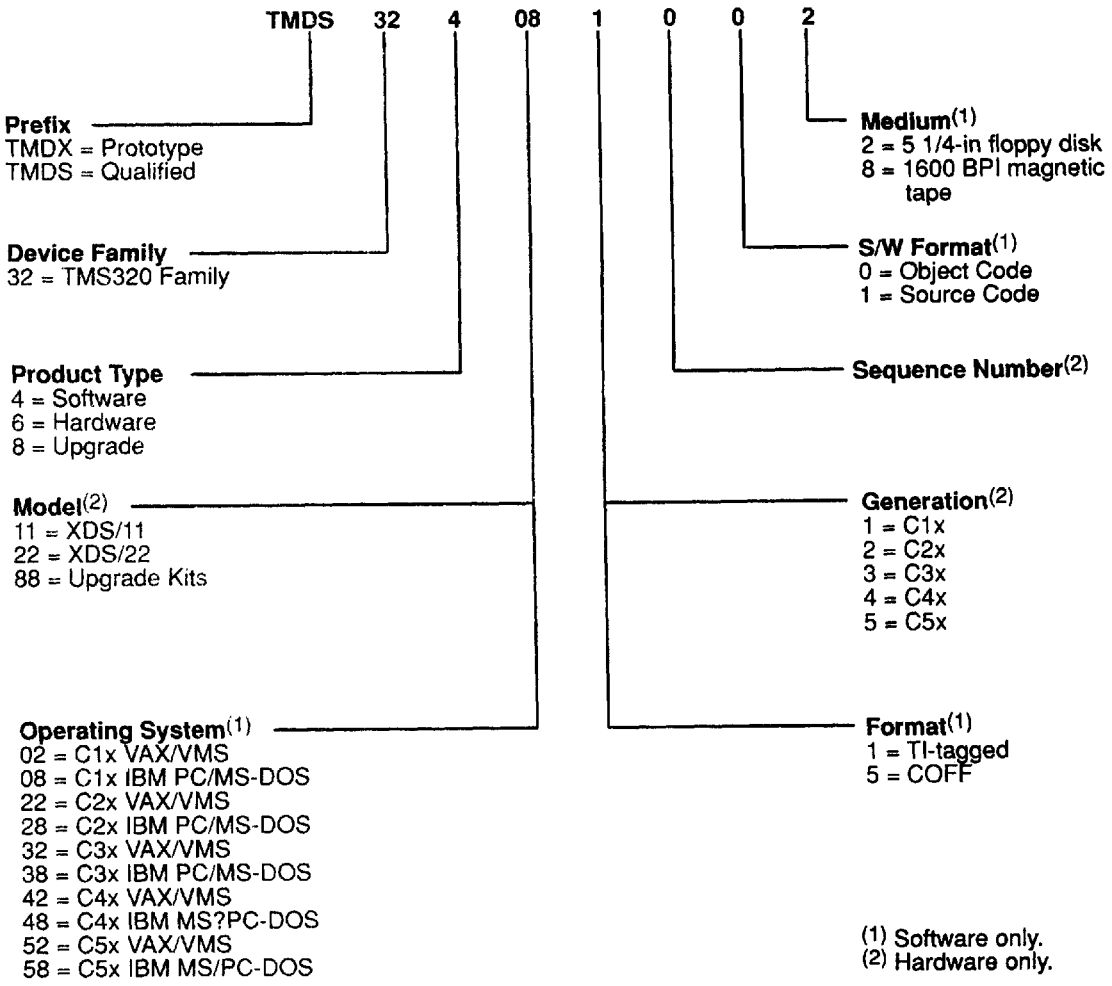


Figure E–2 provides a legend for reading the part number for any TMS320 family software or hardware development tool.

Figure E–2. Development Support Tool Nomenclature



Appendix F

Analog Interface Peripherals and Applications

Texas Instruments offers many different products for total system solutions including various memory options, data acquisition, and analog input/output devices. This appendix describes a variety of devices that interface directly to the TMS320 DSPs in many rapidly expanding applications.

Note:

Throughout this book, 'C14 designates all devices with a 14 suffix (C14/E14/P14), 'C15 all devices with a 15 suffix (C15/E15/LC15/P15), and 'C17 all devices with a 17 suffix (C17/E17/LC17/P17), unless otherwise noted.

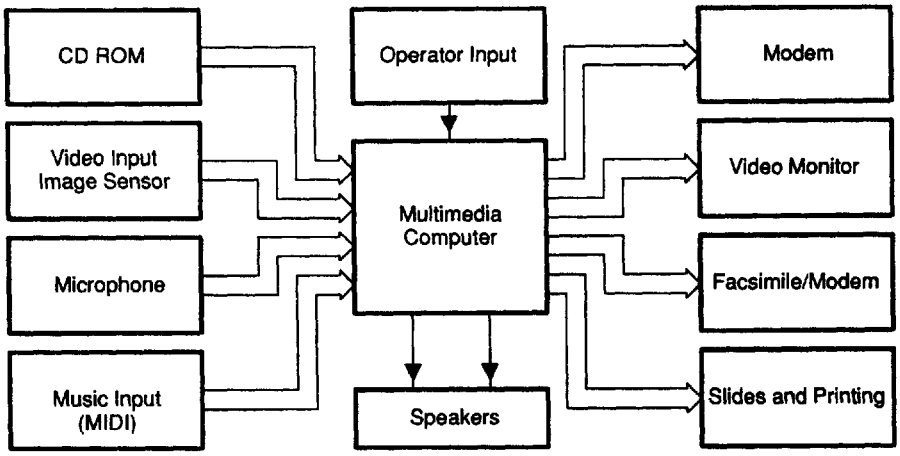
Section **Page**

F.1 Multimedia Applications

Multimedia integrates different media via a centralized computer. These media can be visual or audio and can be input to or output from the central computer via a number of technologies. These technologies can be digital based, such as digital audio and digital video, or analog based, such as audio or video tape recorders. The integration and interaction of media enhances the transfer of information and can accommodate both analysis of problems and synthesis of solutions.

Figure F–1. System Block Diagram

Figure F–1 shows the central role of the multimedia computer and the multimedia system's ability to integrate the various media to optimize information flow and processing.



F.1.1 System Design Considerations

Multimedia systems can include various grades of audio and video quality. The most popular video standard currently used (VGA) covers 640 × 480 pixels with 1, 2, 4, and 8-bit memory-mapped color. Also, 24-bit true color is supported and 1024 × 768 (beyond VGA) resolution is emerging. There are two grades of audio. The lower grade accommodates 11.25 kHz sampling for 8-bit mono, while the higher grade accommodates 44.1-kHz sampling for 16-bit stereo.

Audio specifications include a musical instrument digital interface (MIDI) with compression capability, which is based on keystroke encoding, and an input/output port with a 3-disc voice synthesizer. In the media control area, video disc, CD audio, and CD ROM player interfaces are included. Figure F–2 shows a multimedia subsystem.

The TLC32046 wide-band analog interface circuit (AIC) is well suited for multimedia applications because it features wide-band audio and up to 25-kHz

sampling rates. The TLC32046 is a complete analog-to-digital and digital-to-analog interface system for the TMS320 DSPs. The nominal bandwidths of the filters accommodate 7.6 kHz; however, this bandwidth is programmable. The application circuit shown in Figure F–2 handles both speech encoding and modem communication functions, which are associated with multimedia applications.

Figure F–2. Multimedia Speech Encoding and Modem Communication

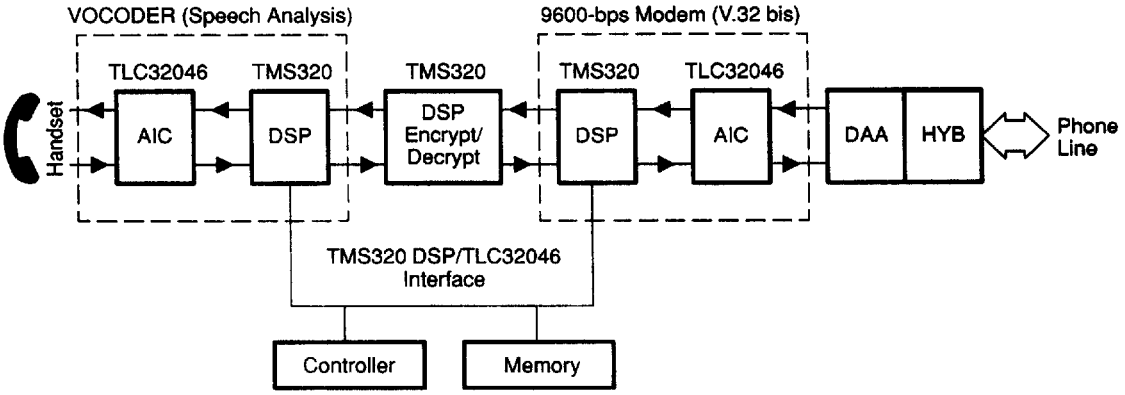
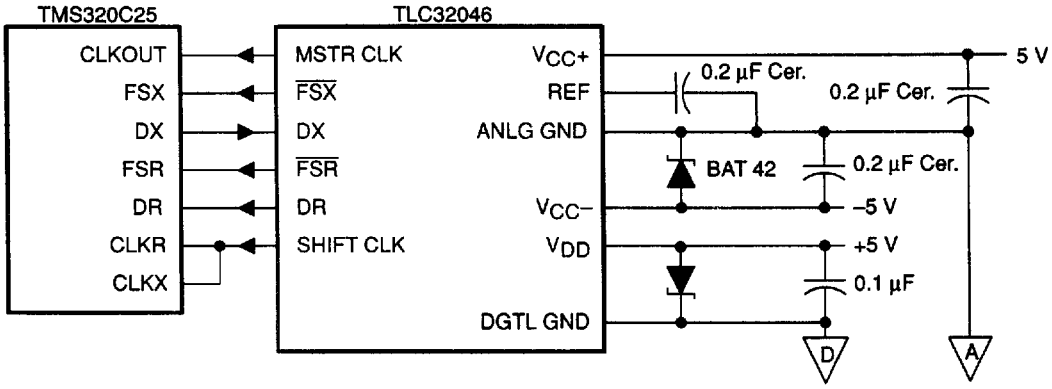


Figure F–3 shows the interfacing of the TMS320C25 DSP to the TLC32046 AIC that constitutes the building blocks of the 9600-bps V.32 bis modem shown in Figure F–2.

Figure F–3. TMS320C25 to TLC32046 Interface



F.1.2 Multimedia-Related Devices

As shown in Table F–1, TI provides a complete array of analog and graphics interface devices. These devices support the TMS320 DSPs for complete multimedia solutions.

Table F–1. Multimedia-Related Devices

Device	Description	I/O	Resolution (Bits)	Conversion CLK Rate	Application
TLC32046	Analog interface (AIC)	Serial	14	25 kHz	Speech and modems
TLC32044	Analog interface (AIC)	Serial	14	19.2 kHz	Speech and modems
TLC32040	Analog interface (AIC)	Serial	14	19.2 kHz	Speech and modems
TLC34075	Video palette	Parallel	Triple 8	135 MHz	Graphics
TLC34058	Video palette	Parallel	Triple 8	135 MHz	Graphics
TLC5502	Flash ADC	Parallel	8	20 MHz	Video
TLC5602	Video DAC	Parallel	8	20 MHz	Video
TLC5501	Flash ADC	Parallel	6	20 MHz	Video
TLC5601	Video DAC	Parallel	6	20 MHz	Video
TLC1550/1	ADC	Parallel	10	150 kHz	Servo ctrl / speech
TLC32070	Analog interface (AIC)	Parallel	8	1 MHz	Servo ctrl / disk drive
TMS57021	Digital filter	Serial	18	44.1 kHz	Digital audio
TMS57010	Dual 18-Bit DAC	Serial	18	44.1 kHz	Digital audio

For application assistance or additional information, please call (214) 997–3772.

F.2 Telecommunications Applications

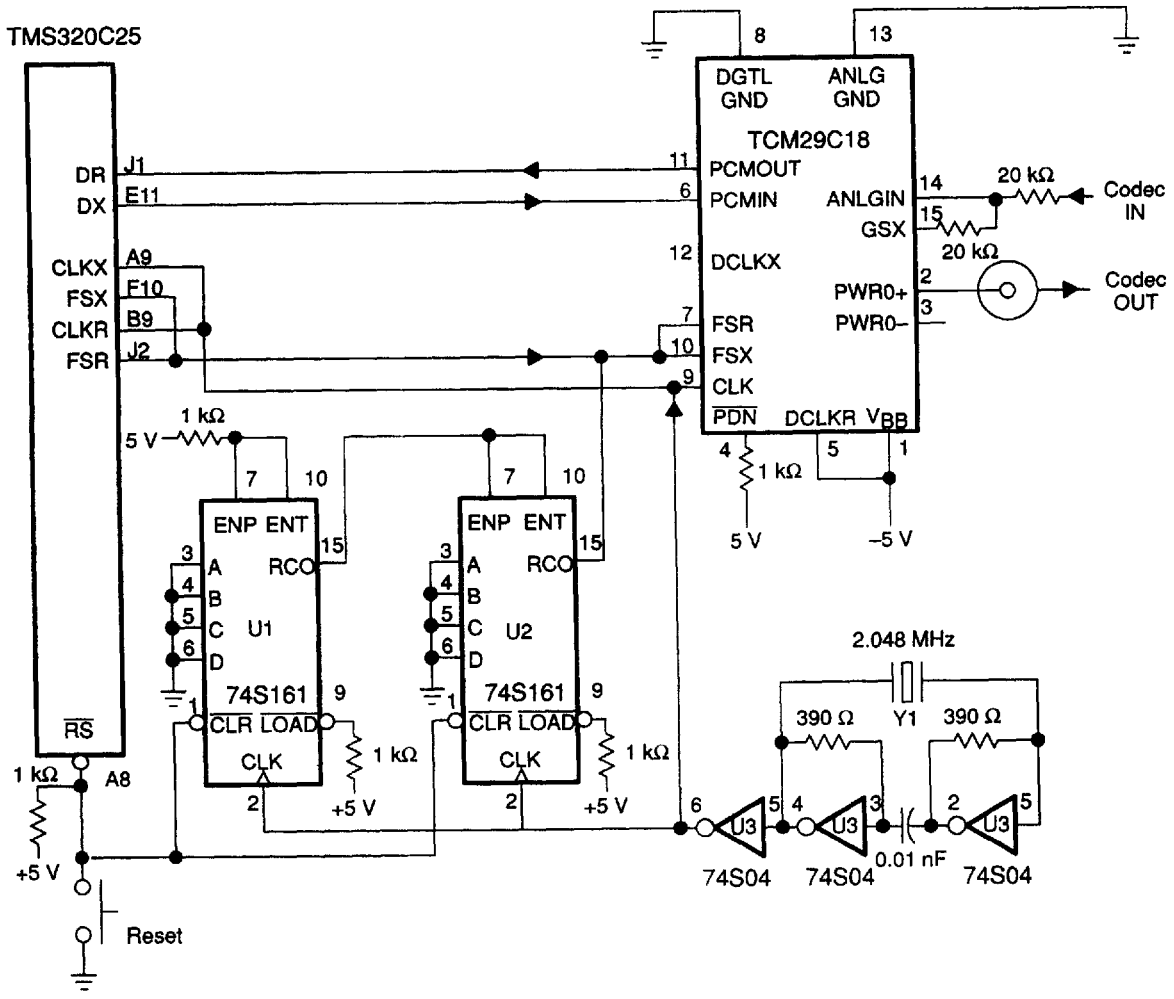
TI's linear product line focuses on two primary telecommunications application areas: subscriber instruments (telephones, modems, etc.) and central office line card products. Subscriber instruments include the TCM508x DTMF tone encoder family, the TCM150x tone ringer family, the TCM1520 ring detector, and the TCM3105 FSK modem. Central office line card products include the TCM29Cxx combo (combined PCM filter plus codec) family, the TCM420x subscriber line control circuit family, and the TCM1030/60 line card transient protector.

TI continues to develop new telecom integrated circuits such as a solid-state subscriber line interface circuit (SLIC) that will significantly outperform today's transformer based designs, and a high-performance 5-volt combo, targeted at cellular phone applications. An RF power amplifier family is aimed at the hand-held and mobile cellular phone market.

F.2.1 System Design Considerations

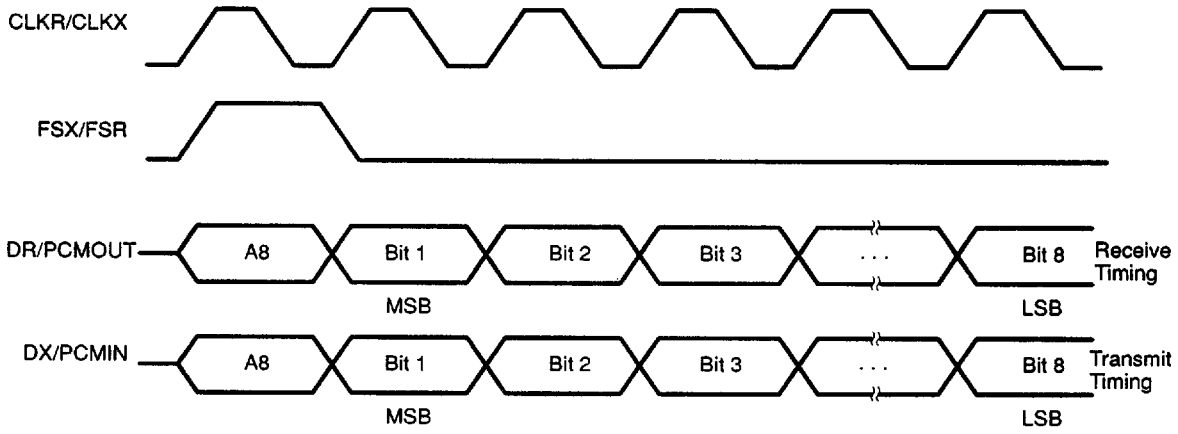
The size, network complexity and compatibility requirements of telecommunications central office systems create very demanding performance requirements. Combo voice-band filter performance is typically ± 0.15 dB in the pass-band. Idle channel noise must be on the order of 15 dBrnc0. Gain tracking (S/Q) and distortion must also meet stringent requirements. The key parameters for a SLIC device are gain, longitudinal balance, and return loss.

Figure F–4. Typical DSP/Combo Interface



The TCM29C18 combo interfaces directly to the TMS320C25 serial port with a minimum of external components, as shown in Figure F–4. Half of hex inverter U3 and crystal Y1 form an oscillator that provides clock timing to the TCM29C18. The synchronous 4-bit counters U1 and U2 generate an 8-kHz frame sync signal. DCLKR on the TCM29C18 is connected to V_{BB}, placing the combo in fixed data-rate mode. Two 20-kΩ resistors connected to ANLGIN and GSX set the gain of the analog input amplifier to 1. The timing is shown in Figure F–5.

Figure F–5. DSP/Combo Interface Timing



F.2.1.1 Telecommunications-Related Devices

Data sheets for the devices in Table F–2 are contained in the *1991 Telecommunications Circuits Databook*, (literature number SCTD001B). To request your copy, contact your nearest Texas Instruments field sales office or call the Customer Response Center at (800) 336–5236.

Table F–2. Telecom Devices

Device Number	Coding Law	Clock Rates MHz†	# of Bits	Comments
Codec/Filter				
TCM29C13	A and μ	1.544, 1.536, 2.048	8	C.O. and PBX line cards
TCM29C14	A and μ	1.544, 1.536, 2.048	8	Includes 8th-bit signal
TCM29C16	μ	2.048	8	16-pin package
TCM29C17	A	2.048	8	16-pin package
TCM29C18	μ	2.048	8	Low-cost DSP interface
TCM29C19	μ	1.536	8	Low-cost DSP interface
TCM29C23	A and μ	Up to 4.096	8	Extended frequency range
TCM29C26	A and μ	Up to 4.096	8	Low-power TCM29C23
TCM320AC36	μ and Linear	Up to 4.096	8 and 13	Single voltage (+5)
TCM320AC37	A and Linear	Up to 4.096	8 and 13	Single voltage (+5)
TP3054/64	μ	1.544, 1.536, 2.048	8	National Semiconductor second source
TP3054/67	A	1.544, 1.536, 2.048	8	National Semiconductor second source
TLC32040/1	Linear	Up to 19.2-kHz sampling	14	For high-dynamic linearity
TLC32044/5	Linear	Up to 19.2-kHz sampling	14	For high-dynamic linearity
TLC32046	Linear	Up to 25-kHz sampling	14	For high-dynamic linearity
Solid-State Line Interface				
TCM9050	μ	1.544	8	Line card codec + filter
TCM9051	Linear	Analog	N/A	Line card SLIC
Transient Suppressor				
TCM1030	Transient suppressor for SLIC-based line card			(30 A max)
TCM1060	Transient suppressor for SLIC-based line card			(60 A max)

† Unless otherwise noted

For further information on these telecommunications products, please call (214) 997–3772.

Figure F–6. General Telecom Applications

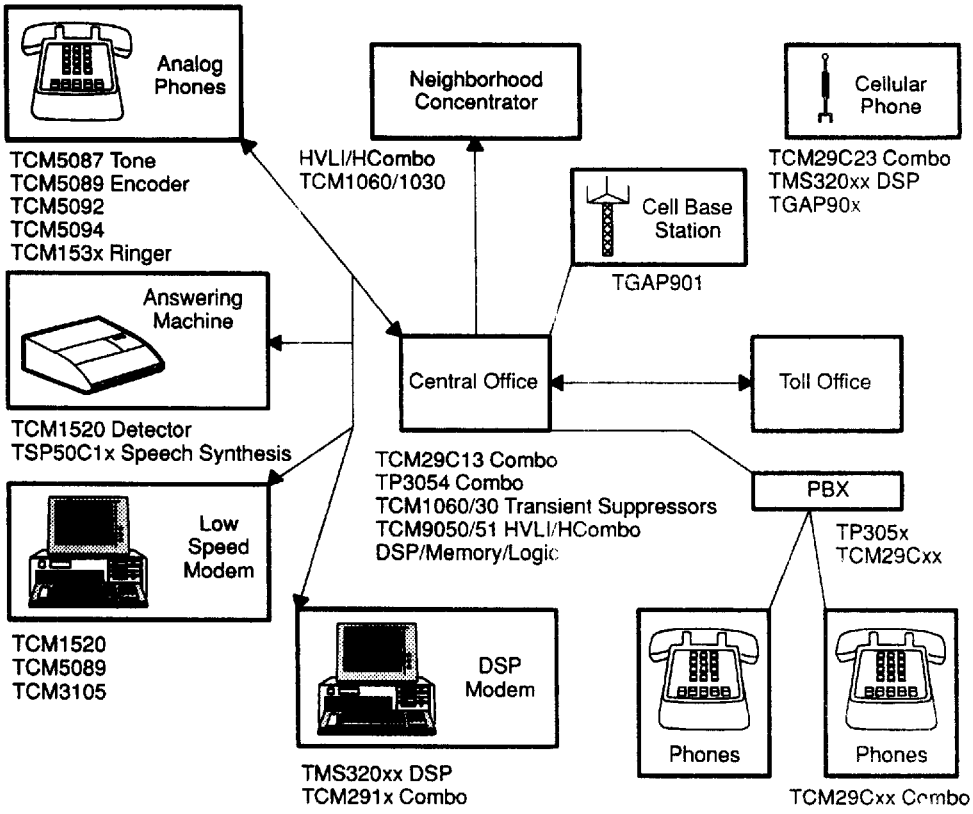
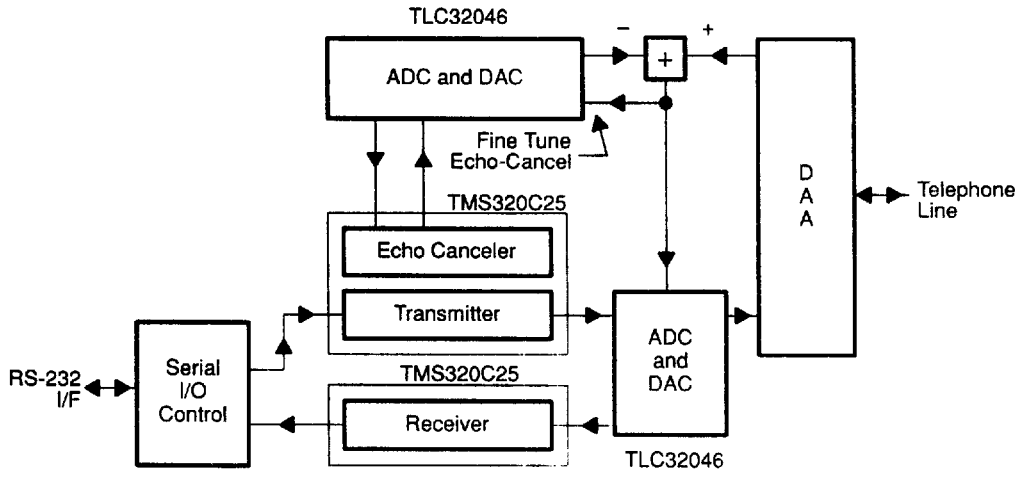


Figure F–7. Generic Telecom Application



F.3 Dedicated Speech Synthesis Applications

For dedicated speech synthesis applications, Texas Instruments offers a family of dedicated speech synthesizer chips. This speech technology has been used in a wide range of products including games, toys, burglar alarms, fire alarms, automobiles, airplanes, answering machines, voice mail, industrial control machines, office machines, advertisements, novelty items, exercise machines, and learning aids.

Dedicated speech synthesis chips are a good alternative for very low cost applications. The speech synthesis technology provided by the dedicated speech synthesizer chips is either LPC (linear-predictive coding) or CVSD (continuously variable slope delta modulation). Table F–3 shows the characteristics of the TI voice synthesizers.

Table F–3. Voice Synthesizers

TI Voice Synthesizers:						
Device	**Microprocessor**	**Synthesis Method**	**I/O Pins**	**On-Chip Memory (Bits)**	**External Memory**	**Data Rate (Bits/Sec)**
TSP50C4x	8-bit	LPC–10	20/32	64K/128K	VROM	1200–2400
TSP50C1x	8-bit	LPC–12	10	64K/128K	VROM	2400
TSP53C30	8-bit	LPC–10	20	N/A	From host µP	1200–2400
TSP50C20	8-bit	LPC–10	32	N/A	EPROM	1200–2400
TMS3477	N/A	CVSD	2	None	DRAM	16K–32K

In addition to the speech synthesizers, TI has low cost memories that are ideal for use with these chips. Texas Instruments can also be of assistance in developing and processing the speech data which is used in these speech synthesis systems. Table F–4 shows speech memory devices of different capabilities.

Table F–4. Speech Memories

TSP60Cxx Family of Speech ROMs					
	TSP60C18	TSP60C19	TSP60C20	TSP60C80	TSP60C81
Size	256K	256K	256K	1M	1M
No. of Pins	16	16	28	28	28
Interface	Parallel 4-bit	Serial	Parallel/serial 8-bit	Serial	Parallel 4-bit
For use with:	TSP50C1x	TSP50C4x	TSP50C4x	TSP50C4x	TSP50C1x

F.3.1 Speech Synthesis Development Tools

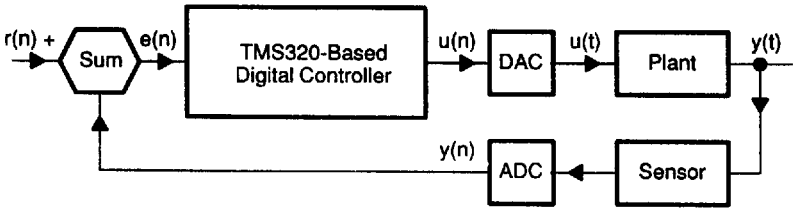
Software:		**System:**	
EVM	Code development tool	SEB	System emulator board
Speech:		SEB60Cxx	System emulator boards for speech memories
SAB	Speech audition board		
SD85000	PC-based speech analysis system		

For further information on these speech synthesis products, please call (214) 997–3772.

F.4 Servo Control/Disk Drive Applications

Several years ago, most servo control systems used only analog circuitry. However, the growth of digital signal processing has made digital control theory a reality. Figure F–8 shows a block diagram of a generic digital control system using a DSP, along with an ADC and DAC.

Figure F–8. Generic Servo Control Loop



In a DSP-based control system, the control algorithm is implemented via software. No component aging or temperature drift is associated with digital control systems. Additionally, sophisticated algorithms can be implemented and easily modified to upgrade system performance.

F.4.1 System Design Considerations

TMS320 DSPs have facilitated the development of high-speed digital servo control for disk drive and industrial control applications. Disk drives have increased storage capacity from 5 megabytes to over 1 gigabyte in the past decade, which equates to a 23,900 percent growth in capacity. To accommodate these increasingly higher densities, the data on the servo platters, whether servo-positioning or actual storage information, must be converted to digital electronic signals at increasingly closer points in relation to the platter "pick-off" point. The ADC must have increasingly higher conversion rates and greater resolution to accommodate the increasing bandwidth requirements of higher storage densities. In addition, the ADC conversion rates must increase to accommodate the shorter data retrieval access time.

Figure F–9 shows a block diagram of a disk drive control system.

Figure F–9. Disk Drive Control System Block Diagram

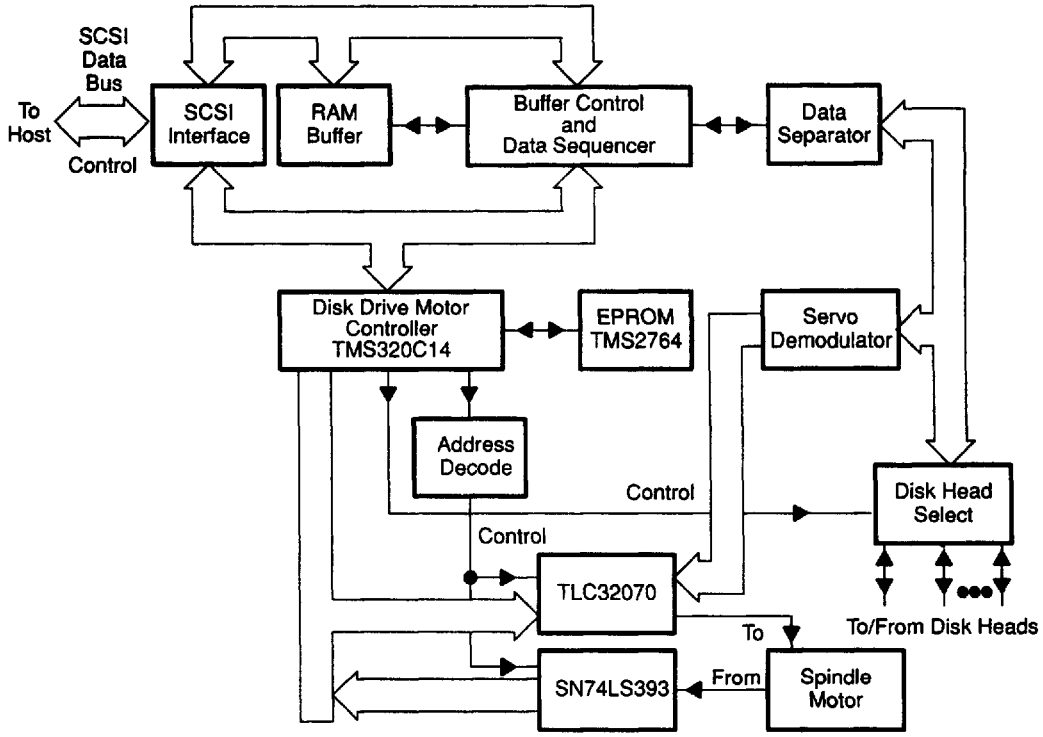


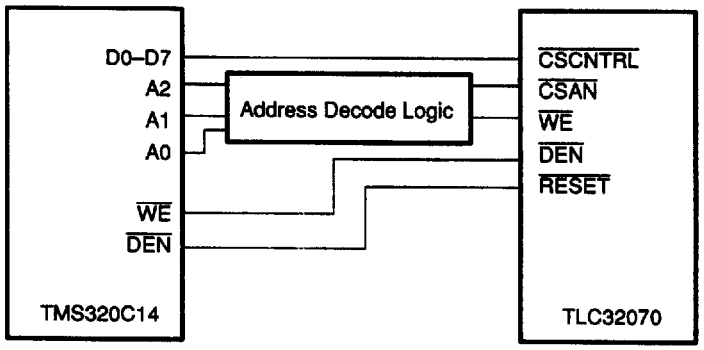
Table F–5 lists analog/digital interfacing devices used for servo control.

Table F–5. Servo Control Related Devices

Function	Device	Bits	Speed	Channels	Interface
ADC	TLC1550	10	3–5 µs	1	Parallel
	TLC1551	10	3–5 µs	1	Parallel
	TLC5502	8	50 ns (flash)		Parallel
	TLC0820	8	1 µs	1	Parallel
	TLC1225	13	12 µs	1 (Diff.)	Parallel
	TLC1125	13	12 µs	1 (Diff.)	Parallel
	TLC1558	10	3–5 µs	8	Parallel
	TLC1542	10	3–5 µs	11	Serial
DAC	TLC7524	8	9 MHz	1	Parallel
	TLC7628	8	9 MHz	(Dual)	Parallel
	TLC5602	8	30 MHz	1	Parallel
AIC	TLC32070	8 (ADC)	1 µs	8	Parallel
		8 (ADC)	9 MHz	1	Parallel

Figure F–10 shows the interfacing of the TMS320C14 and the TLC32070.

Figure F–10. TMS320C14 – TLC32070 Interface



For further information on these servo control products, please call (214) 997–3772.

F.5 Modem Applications

High-speed modems (9,600 bps and above) require a great deal of analog signal processing in addition to digital signal processing. Designing both high-speed capabilities and slower fall-back modes poses significant engineering challenges. TI offers a number of analog front end (AFE) circuits to support various modem standards.

The TLC32040, TLC32044, and TLC32046 analog interface circuits (AIC) are specially suited for modem applications by the integration of an input multiplexer, switched capacitor filters, high resolution 14-bit ADC and DAC, a four-mode serial port, and control and timing logic. These converters feature adjustable parameters such as filtering characteristics, sampling rates, gain selection, (sin x)/x correction (TLC32044 and TLC32046 only), and phase adjustment. All these parameters are software programmable, making the AIC suitable for a variety of applications. Table F–6 has the description and characteristics of these devices.

Table F–6. Modem AFE Data Converters

Device	Description	I/O	Resolution (Bits)	Conversion Rate
TLC32040	Analog interface chip (AIC)	Serial	14	19.2 kHz
TLC32041	AIC without on-board V_{REF}	Serial	14	19.2 kHz
TLC32042	AIC with 200-Hz roll-off	Serial	14	19.2 kHz
TLC32044	Telephone speed/modem AIC	Serial	14	19.2 kHz
TLC32045	Low-cost version of the TLC32044	Serial	14	19.2 kHz
TLC32046	Wide-band AIC	Serial	14	25 kHz
TCM29C18	Companding codec/filter	PCM	8	8 kHz
TCM29C23	Companding codec/filter	PCM	8	16 kHz
TCM29C26	Low-power codec/filter	PCM	8	16 kHz
TCM320AC36	Single-supply codec/filter	PCM and Linear	8	25 kHz

The AIC interfaces directly with serial-input TMS320 DSPs, which execute the modem's high-speed encoding and decoding algorithms. The TLC3204x family performs level-shifting, filtering and A/D/A data conversion. The DSP's many software-programmable features provide the flexibility required for modem operations and make it possible to modify and upgrade systems easily. Under DSP control, the AIC's sampling rates permit designers to include fall-back modes without additional analog hardware in most cases. Phase adjustments can be made in real time so that the A/D and D/A conversions can be synchronized with the upcoming signal. In addition, the chip has a built-in loopback feature to support modem self-test requirements.

For further information or application assistance, please call (214) 997–3772.

Figure F–11. High-Speed V.32 and Multistandard Modem With the TLC32046 AIC

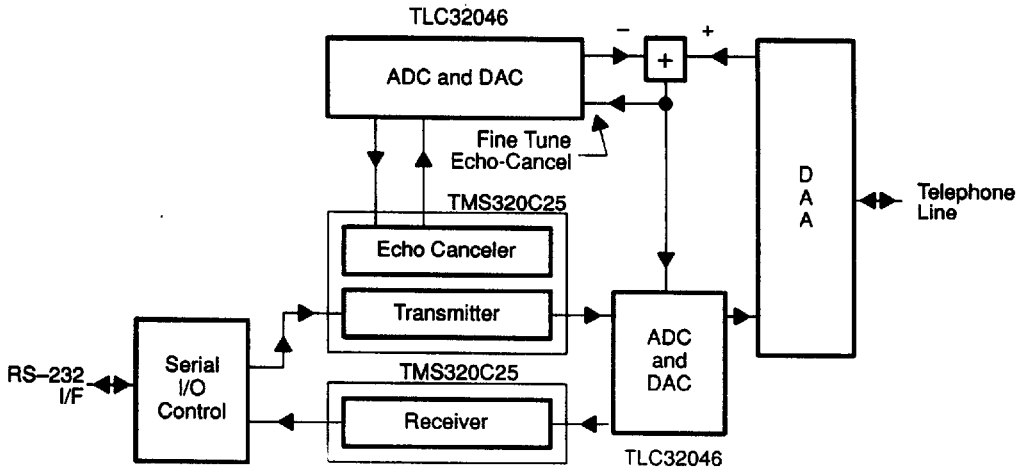


Figure F–11 shows a V.32 modem implementation using the TMS320C25 and a TLC32046. The upper TMS320C25 performs echo cancellation and transmit data functions while the lower TMS320C25 performs receive data and timing recovery functions. The echo canceler simulates the telephone channel and generates an estimated echo of the transmit data signal. The TLC32046 performs the following functions:

Upper TLC32046 D/A Path: Converts the estimated echo, as computed by the upper TMS320C25, into an analog signal, which is subtracted from the receive signal.

Upper TLC32046 A/D Path: Converts the residual echo to a digital signal for purposes of monitoring the residual echo and continuously training the echo canceler for optimum performance. The converted signal is sent to the upper TMS320C25.

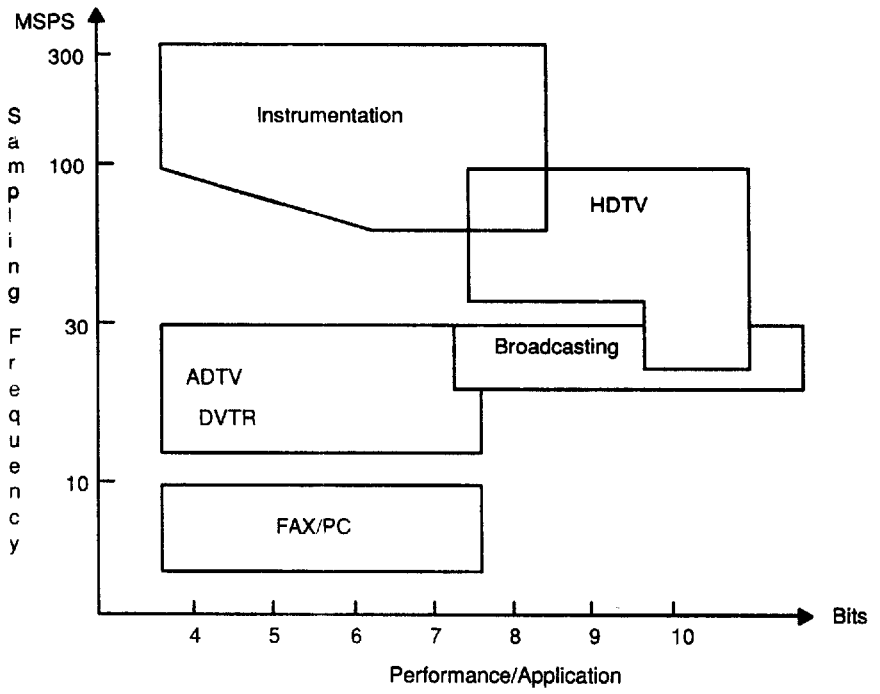
Lower TLC32046 D/A Path: Converts the upper TMS320C25 transmit output to an analog signal, performs a smoothing filter function, and drives the DAC.

Lower TLC32046 D/A Path: Converts the echo-free receive signal to a digital signal, which is sent to the lower TMS320C25 to be decoded.

F.6 Advanced Digital Consumer Electronics Applications

With the extensive use of the TMS320 DSPs in consumer electronics, much of the electromechanical control and signal processing can be done in the digital domain. Digital systems generally require some form of analog interface, usually in the form of high-performance ADCs and DACs. Figure F–12 shows the general performance requirements for a variety of applications.

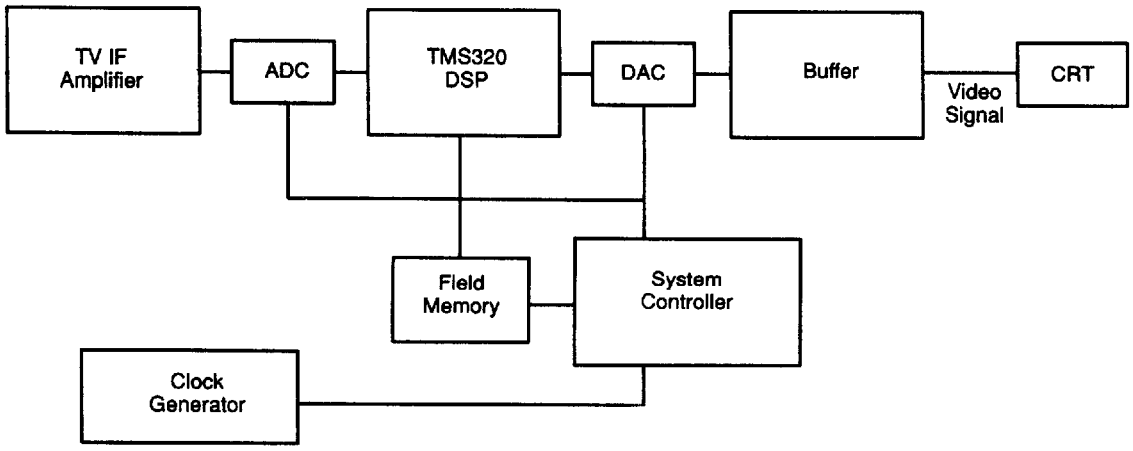
Figure F–12. Applications Performance Requirements



F.6.1 Advanced Television System Design Considerations

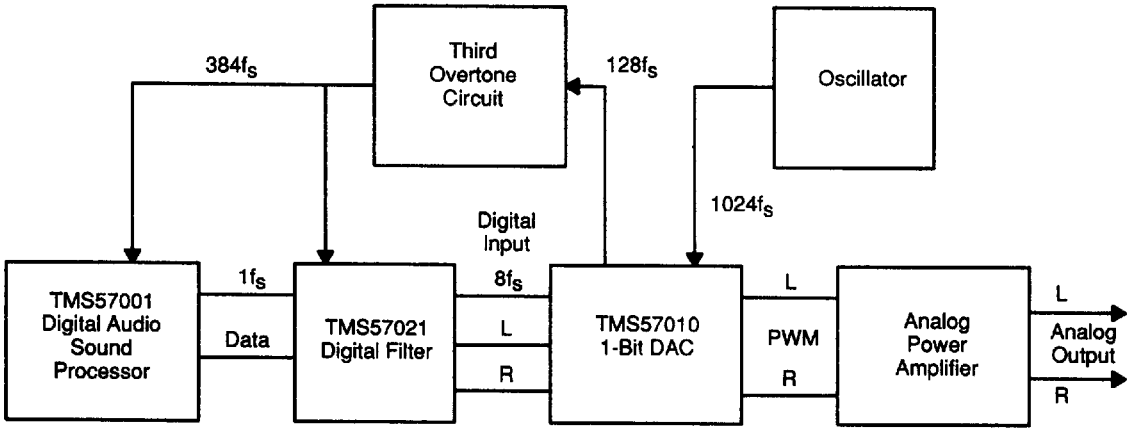
Advanced Digital Television (ADTV) is a technology that uses digital signal processing to enhance video and audio presentations and to reduce noise and ghosting. Because of these DSP techniques, a variety of features can be implemented, including frame store, picture-in-picture, improved sound quality, and zoom. The bandwidth requirements remain at the existing 6-MHz television allocation. From the IF(intermediate frequency) output, the video signal is converted by an 8-bit video ADC. The digital output can be processed in the digital domain to provide noise reduction, interpolation or averaging for digitally increased sharpness, and higher quality audio. The DSP digital output is converted back to analog by a video DAC, as shown in Figure F–13.

Figure F–13. Video Signal Processing Basic System



VCRs, compact disc and DAT players, and PCs are a few of the products that have taken a major position in the marketplace in the last ten years. The audio channels for compact disc and DAT require 16-bit A/D resolution to meet the distortion and noise standards. See Figure F–14 for a block diagram of a typical digital audio system.

Figure F–14. Typical Digital Audio Implementation



The motion and motor control systems usually use 8- to 10-bit ADCs for the lower frequency servo loop. Tape or disc systems use motor or motion control for proper positioning of the record or playback heads. With the storage medium compressing data into an increasingly smaller physical size, the positioning systems require more precision

The audio processing becomes more demanding as higher fidelity is required. Better fidelity translates into lower noise and distortion in the output signal.

The TMS57010 is a 1-bit oversampling DAC designed for digital audio systems, such as CD players, DATs, digital amplifiers, and broadcast satellite tuners. It contains two serially loaded 18-bit multiplying, pulse-width modulated (PWM) DACs. A third-order MASH (multistage noise-shaping) filter technology is featured. It is suitable for 8× oversampling digital filters, such as the TMS57020 and TMS57021.

The TMS57021 is a digital filter LSI with 4×/8× oversampling rates. The TMS57021 features noise shaping, attenuation, soft muting, de-emphasis, and a wide variety of built-in functions. Ripple is within 10^{-5} dB, and attenuation is greater than –100 dB at 24.1 kHz.

Table F–7 lists some of TI's products for analog interfacing to digital systems.

Table F–7. Audio/Video Analog/Digital Interface Devices

Function	Device	Bits	Speed	Channels	Interface
D/A (PWM)	TMS57010	18 (Equiv.)	44.1 kHz	2	Serial
Digital filter (10^{-5} dB ripple, -10^{2} dB attenuation)	TMS57021	N/A	44.1 kHz	2	Serial
Analog interface A/D D/A	TLC32070	 8 8	 2 µs 15 µs	 8 1	 Parallel Parallel
A/D	TLC1225	12	12 µs	1	Parallel
A/D	TLC1550	10	6 µs	1	Parallel
Flash A/D	TLC5503-5	8	0.1 µs	1	Parallel
Video D/A	TLC5602	8	50 ns	1	Parallel
Video D/A	TL5602	8	50 ns	1	Parallel
Flash A/D	TLC5503-2	8	50 ns	1	Parallel
Flash A/D	TLC5502-5	8	50 ns	1	Parallel

For further information or application assistance, please call (214) 997–3772.

Appendix G

Programming the EPROM Cell

This appendix describes the programming, verifying, reading, and protecting of the EPROM cells in the TMS320E1x/P1x devices, which are part of the TMS320C1x digital signal processors generation. The 4K × 16-bit EPROMs are implemented from a standard EPROM cell to expand the capabilities of the TMS320E1x/P1x devices in the areas of prototyping, early field testing, and limited production. For a final production design, the 4K-word masked ROM in the TMS320C1x devices provides a data migration path for cost-effective volume production.

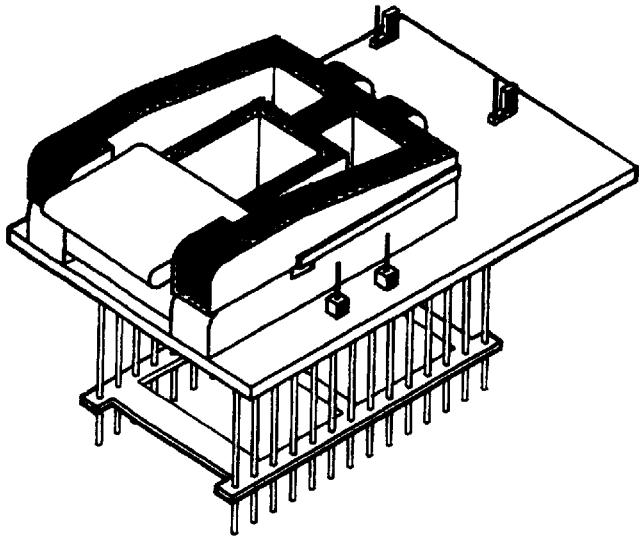
Key features of the EPROM cell include standard programming and verification. The EPROM cell also includes a code protection feature that allows code to be protected against copyright violations. The protection feature can be used to protect reading the EPROM contents.

This appendix contains the following major topics:

G.1 EPROM Adapter Sockets

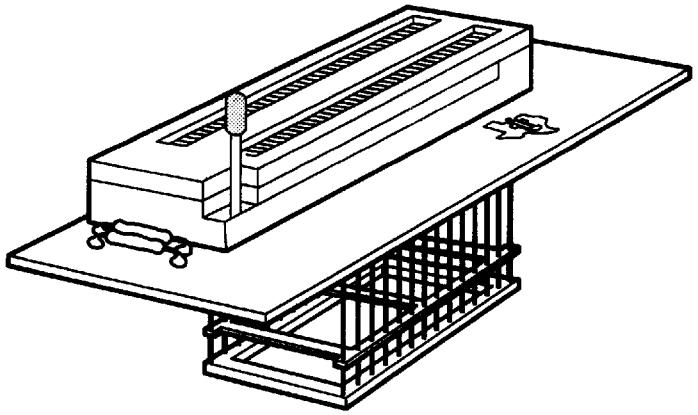
An EPROM adapter socket, shown in Figure G–1, is available to provide 68-pin to 28-pin conversion for programming the TMS320E14/P14.

Figure G–1. EPROM Adapter Socket (68-pin to 28-pin)



EPROM adapter sockets for pin-to-pin conversion of other TMS320E1x/P1x devices are also available. The adapter socket shown in Figure G–2 allows connection of a 40-pin device into an equivalent 28-pin socket.

Figure G–2. EPROM Adapter Socket (40-pin to 28-pin)



G.2 Programming and Verification

The TMS320E1x/P1x EPROM cells are programmed using the same family and device codes as the TMS27C64 8K × 8-bit EPROM. The TMS27C64 EPROM series are ultraviolet-light erasable, electrically programmable read-only memories, fabricated using HVCMOS technology. The TMS27C64 is pin-compatible with existing 28-pin ROMs and EPROMs. The TMS320E1x/P1x devices, like the TMS27C64, operate from a single 5-V supply in the read mode and need an additional 12.5-V supply for programming. All programming signals are TTL level. For programming outside the system, existing EPROM programmers can be used. Locations may be programmed singly, in blocks, or at random.

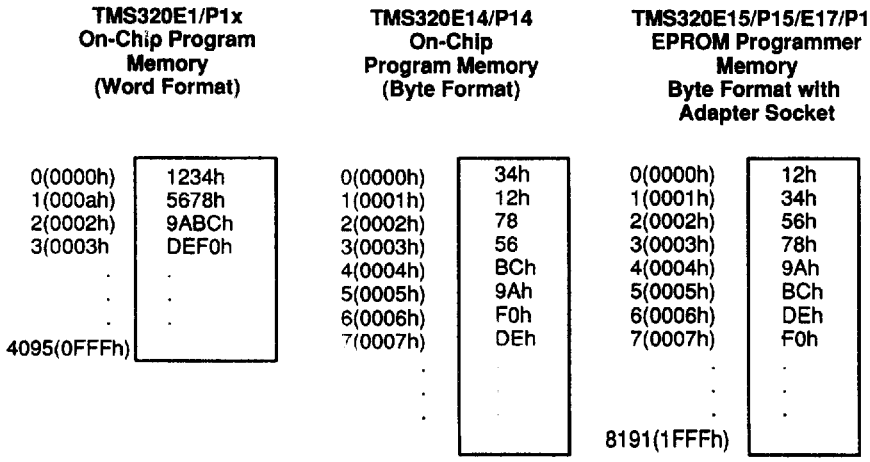
TMS320E14/P14

When programmed in blocks, the data is loaded into the EPROM cell one byte at a time, the low byte first and the high byte second. The 'E14/P14 EPROM adapter socket shown in Figure G–1 has an inverter in it, so that from the EPROM programmer's point of view, data is loaded high byte, then low byte (as shown in Figure G–3). The device itself, however, expects data low-byte first, high-byte second.

TMS320E15/P15/E17/P17

These devices are programmed high-byte first, low-byte second, as shown in Figure G–3.

Figure G–3. EPROM Programming Data Format



The TMS320E1x/P1x devices do not support the signature mode available in some EPROM programmers. The signature mode puts a high voltage (12.5 VDC) on pin A9. The TMS320E1x/P1x EPROM cells are not designed for this feature and will be damaged if subjected to it. A 3.9k ohm resistor is standard on the TI programmer socket between pin A9 and the programmer. This protects the device from unintentional use of the signature mode

Figure G–4 shows the wiring conversion to program the TMS320E14/P14 using the 28-pin pinout of the TMS27C64. Table G–1 shows the pin nomenclature of the TMS320E1x/P1x devices. The code to be programmed into the device should be in serial mode.

Figure G–4. TMS320E14/P14 EPROM Conversion to TMS27C64 EPROM

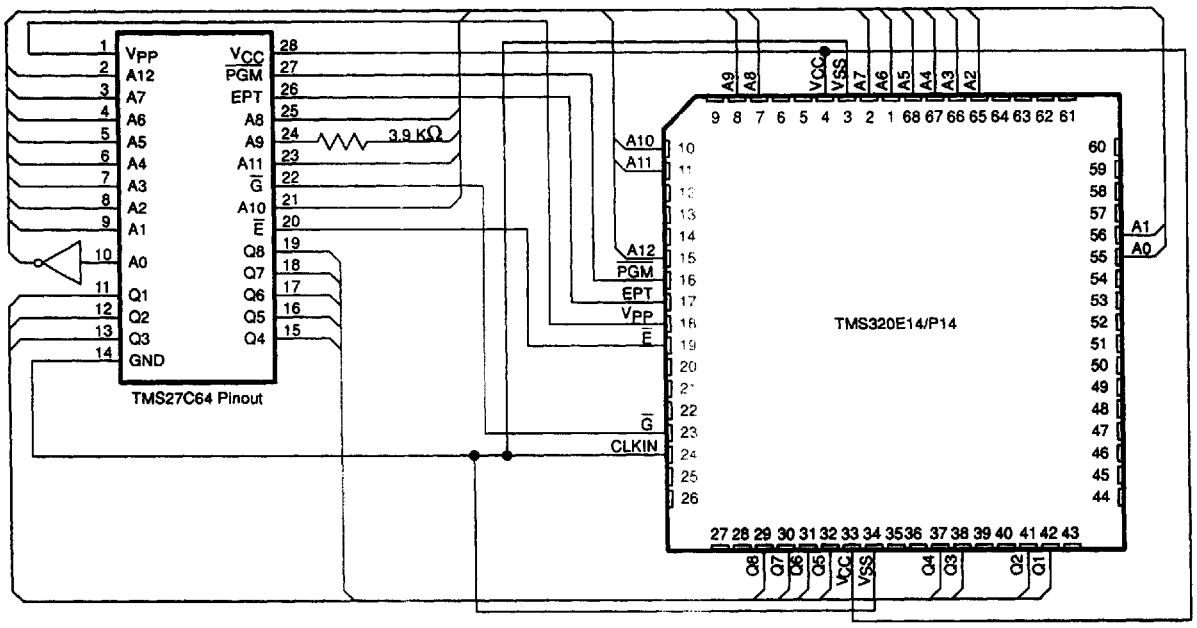


Table G–1. TMS320E1x/P1x Pin Nomenclature

Name	I/O	Definition
A12(MSB)–A0(LSB)	I	On-chip EPROM programming address lines
CLKIN	I	Clock oscillator input
$\overline{E}$	I	EPROM chip enable
EPT	I	EPROM test mode select
$\overline{G}$	I	EPROM output enable
GND	I	Ground
$\overline{PGM}$	I	EPROM write/program select
Q8(MSB)–Q1(LSB)	I/O	Data lines for byte-wide programming of on-chip 8K bytes of EPROM
$\overline{RS}$	I	Reset for initializing the device
V_{CC}	I	5-V power supply
V_{PP}	I	12.5-V programming power supply

Figure G–5. TMS320E15/P15/E17/P17 EPROM Conversion to TMS27C64 EPROM

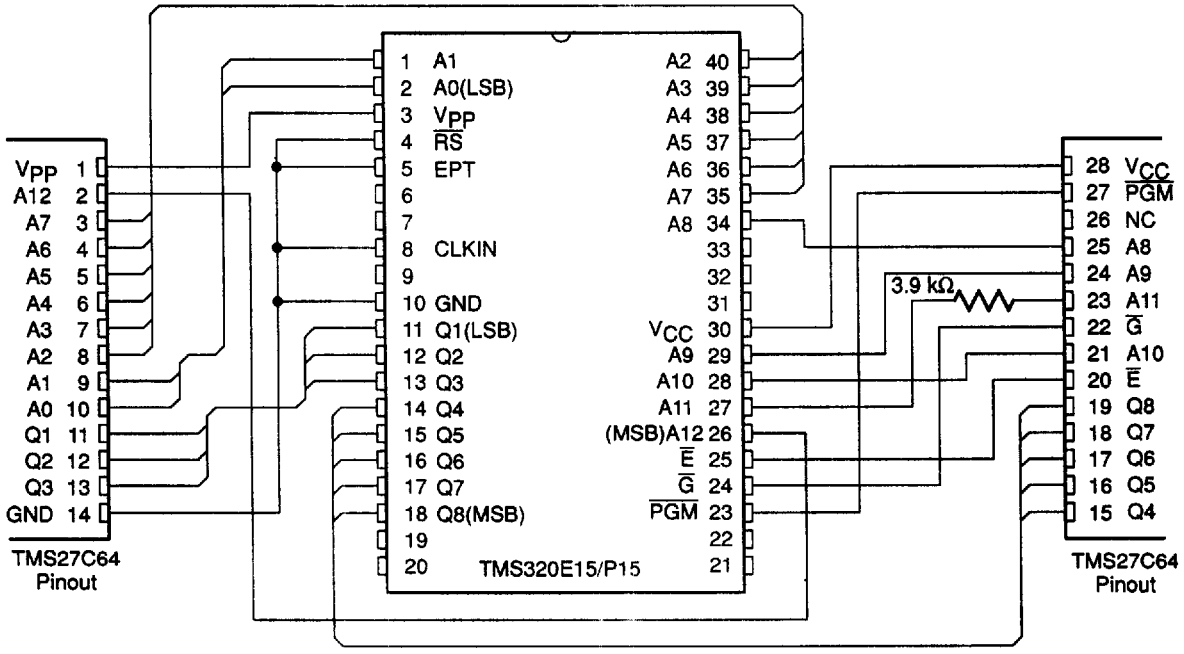


Figure G–5 shows the wiring conversion to program the 'E15/P15/E17/P17 using the 28-pin pinout of the TMS27C64. The code to be programmed into the device should be in serial mode.

Figure G–6 shows the programming levels required for programming, verifying, and reading the EPROM cell. The paragraphs following the table describe the function of each programming level.

Figure G–6. TMS320E1x/P1x Programming Mode Levels

Signal Name	'E15/P15 'E17/P17 DIP Pin	'E14/P14 Pin	TMS27C64 Pin	Program	Program Verify	Program Inhibit	Read	Output Disable
$\overline{E}$	25	19	20	V_{IL}	V_{IL}	V_{IH}	V_{IL}	V_{IL}
$\overline{G}$	24	23	22	V_{IH}	$\overline{PULSE}$	X	$\overline{PULSE}$	V_{IH}
$\overline{PGM}$	23	16	27	$\overline{PULSE}$	V_{IH}	V_{IH}	V_{IH}	V_{IH}
V_{PP}	3	18	1	V_{PP}	V_{PP}	V_{PP}	V_{CC}	V_{CC}
V_{CC}	30	4,33	28	V_{CCP}	V_{CCP}	V_{CCP}	V_{CC}	V_{CC}
V_{SS}	10	3,34	14	V_{SS}	V_{SS}	V_{SS}	V_{SS}	V_{SS}
CLKIN	8	24	14	V_{SS}	V_{SS}	V_{SS}	V_{SS}	V_{SS}
$\overline{RS}$	4	–	14	V_{SS}	V_{SS}	V_{SS}	V_{SS}	V_{SS}
EPT	5	17	26	V_{SS}	V_{SS}	V_{PP}	V_{SS}	V_{SS}
Q8–Q1	18–11	29–32,37, 38,41,42	19–15,13–11	D_{IN}	Q_{OUT}	HI-Z	Q_{OUT}	HI-Z
A12–A7	26–28,29, 34,35	15,11,10,8, 7,2	2,23,21,24, 25,3	ADDR	ADDR	X	ADDR	X
A6	36	1	4	ADDR	ADDR	X	ADDR	X
A5	37	68	5	ADDR	ADDR	X	ADDR	X
A4	38	67	6	ADDR	ADDR	X	ADDR	X
A3–A0	39,40,1,2	66,65,56,55	7–10	ADDR	ADDR	X	ADDR	X

Note: See Appendix A for TMS320E1x/P1x pinouts.

LEGEND:
V_{IH} = TTL high level
V_{IL} = TTL low level
ADDR = byte address bit
V_{PP} = 12.5 ± 0.25 V (FAST)
or 13 ± 0.25 V (SNAP!, TMS302E14/P14 only)
V_{CC} = 5 ± 0.25 V
V_{CCP} = 6 ± 0.25 V (FAST)
or 6.5 ± 0.25 V for (SNAP!, TMS302E14/P14 only)
X = don't care
PULSE = low-going TTL pulse
DIN = byte to be programmed at ADDR
Q_{OUT} = byte stored at ADDR

G.2.1 Erasure

Before programming, the device is erased by exposing the chip through the transparent lid to high-intensity ultraviolet light (wavelength 2537 angstroms). The recommended minimum exposure dose (UV-intensity × exposure-time) is 15 watt-seconds per square centimeter. A typical 12 milliwatt per square centimeter, filterless UV lamp will erase the device in 21 minutes. The lamp should be located about 2.5 centimeters above the chip during erasure. After erasure, all bits are in the high state. Note that normal ambient light contains the correct

wavelength for erasure. Therefore, when using the TMS320E1x, the window should be covered with an opaque label.

G.2.2 FAST Programming

After erasure (all memory bits in the cell are a logic one), logic zeros are programmed into the desired locations. The FAST programming algorithm, shown in Figure G–7, is normally used to program the entire EPROM contents, although individual locations may be programmed separately. A programmed logic zero can only be erased by ultraviolet light (on TMS320E1x devices). Data is presented in parallel (eight bits) on pins Q8-Q1. Once addresses and data are stable, $\overline{PGM}$ is pulsed. The programming mode is achieved when V_{PP} = 12.5 V, $\overline{PGM} = V_{IL}$, V_{CC} = 6.0 V, $\overline{G} = V_{IH}$, and $\overline{E} = V_{IL}$. More than one TMS320E1x/P1x can be programmed when the devices are connected in parallel. Locations can be programmed in any order.

FAST programming uses two types of programming pulses: prime and final. The length of the prime pulse is 1 ms. After each prime pulse, the byte being programmed is verified. If correct data is read, the final programming pulse is applied; if correct data is not read, an additional 1-ms prime pulse is applied up to a maximum of 25 times. The final programming pulse is 3X times the number of prime programming pulses applied. This sequence of programming and verification is performed at V_{CC} = 6.0 V, and V_{PP} = 12.5 V. When the full FAST programming routine is complete, all bits are verified with $V_{CC} = V_{PP}$ = 5 V.

G.2.3 SNAP! Pulse Programming (TMS320E14/P14)

The EPROM can be programmed using the TI SNAP! pulse programming algorithm illustrated by the flowchart of Figure G–8, which can reduce programming time to a nominal of one second. Actual programming time will vary as a function of the programmer used. Data is presented in parallel (eight bits) on pins Q8 through Q1. Once addresses and data are stable, $\overline{PGM}$ is pulsed.

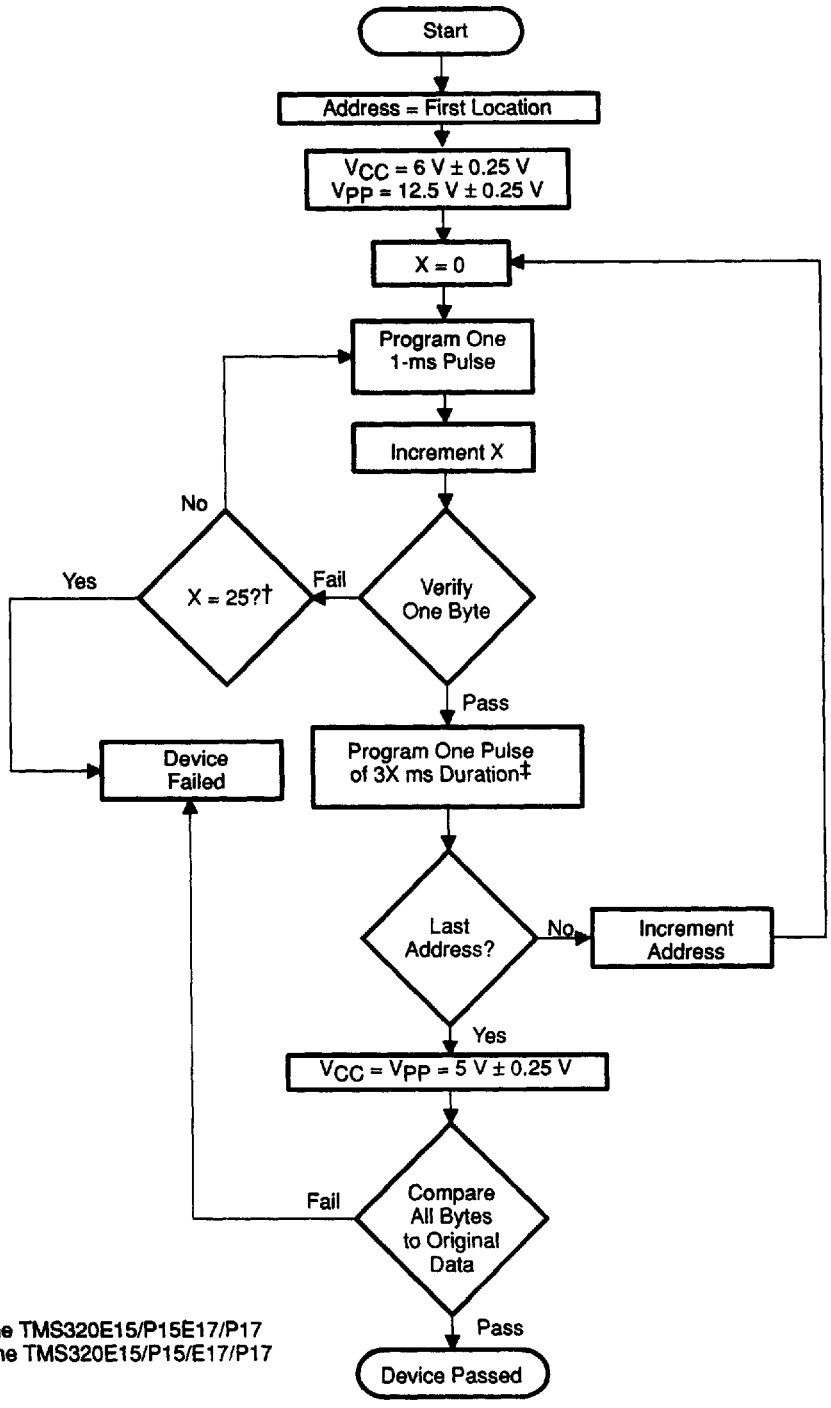
The SNAP! pulse programming algorithm uses pulses of 100 microseconds followed by a byte verification to determine when the addressed byte has been successfully programmed. Up to ten 100-microsecond pulses per byte are provided before a failure is recognized.

The programming mode is achieved when V_{PP} = 13.0 V, V_{CC} = 6.5 V, V and $\overline{G} = V_{IH}$, and $\overline{E} = V_{IL}$. More than one device can be programmed when the devices are connected in parallel. Locations can be programmed in any order. When the SNAP! Pulse programming routine is complete, all bits are verified with $V_{CC} = V_{PP}$ = 5 V.

G.2.4 Program Verify

Programmed bits may be verified with V_{PP} = 12.5 V when $\overline{G} = V_{IL}$, $\overline{E} = V_{IL}$, and $\overline{PGM} = V_{IH}$. Figure G–9 shows the program and verify operations timing for the FAST and for the SNAP! pulse programming.

Figure G–7. FAST Programming Flowchart



† X =15 for the TMS320E15/P15E17/P17
‡ 4X ms for the TMS320E15/P15/E17/P17

Figure G–8. SNAP! Pulse Programming Flowchart (TMS320E14/P14)

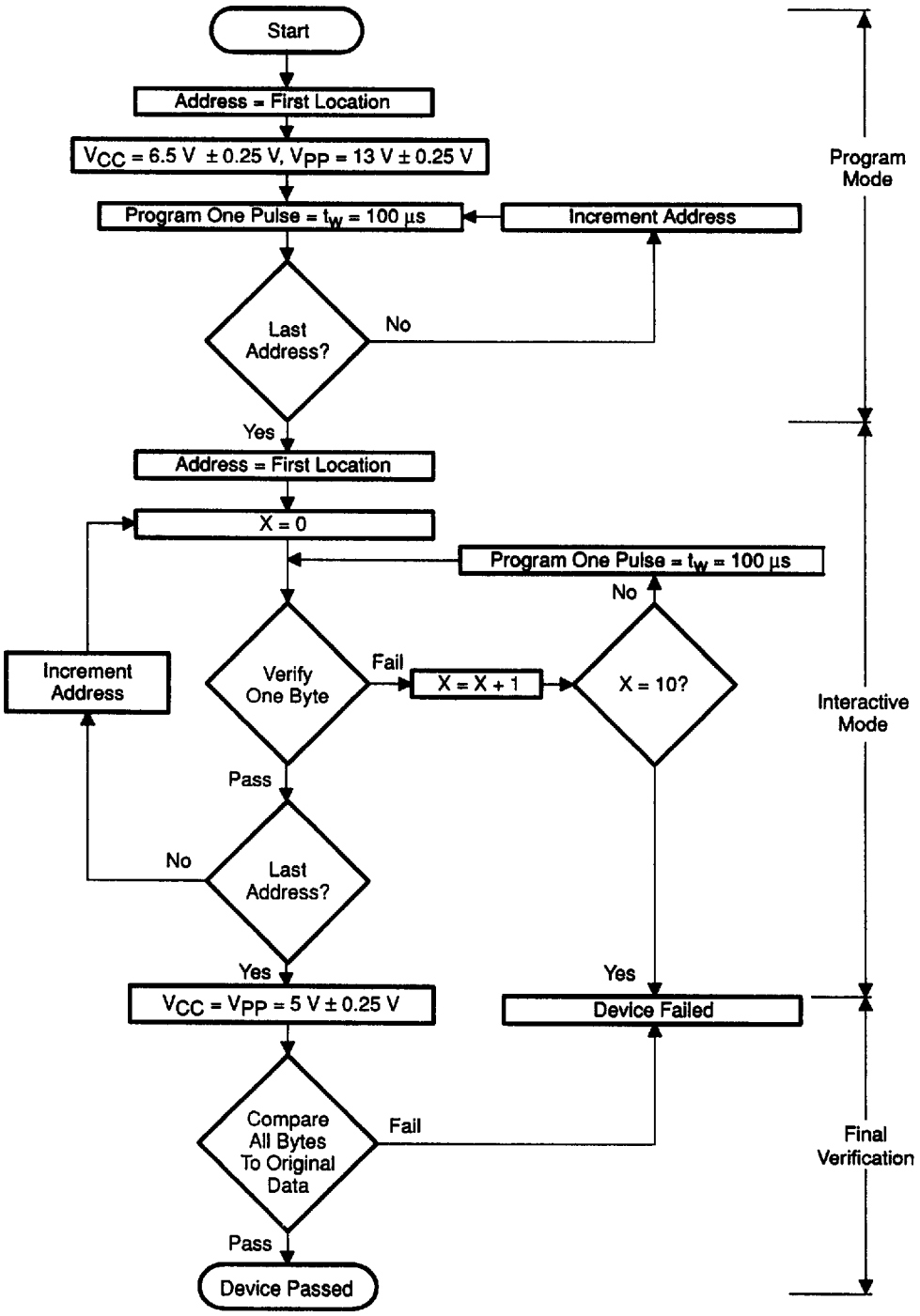
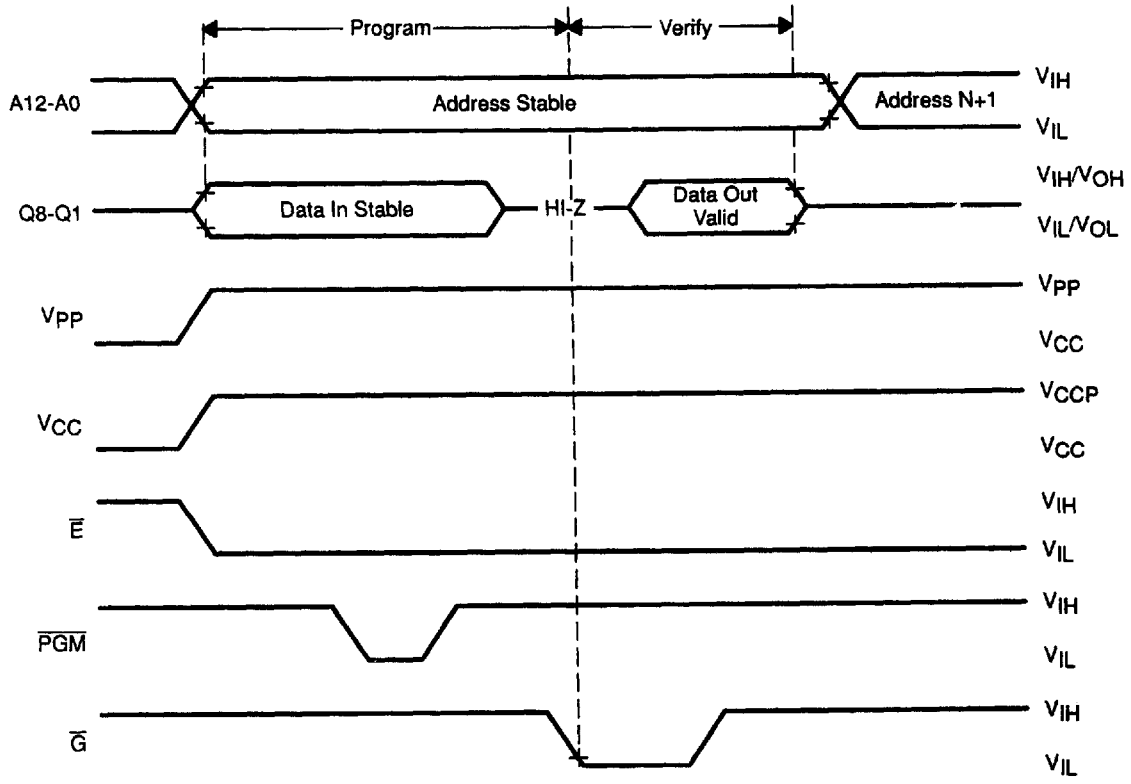


Figure G–9. FAST Programming Timing



G.2.5 Program Inhibit

Programming may be inhibited by maintaining a high-level input on the $\overline{E}$ pin or $\overline{PGM}$ pin.

G.2.6 Read

The EPROM contents may be read independently of the programming cycle, provided the RBIT (ROM protect bit) has not been programmed. The read is accomplished by setting $\overline{E}$ to zero and pulsing $\overline{G}$ low. The contents of the EPROM location, selected by the value on the address inputs, appear on Q8–Q1.

G.2.7 Output Disable

During the EPROM programming process, the EPROM data outputs may be disabled, if desired, by establishing the output disable state. This state is selected by setting the $\overline{G}$ pin high or $\overline{E}$ pin high. Whichever occurs first will determine the time to high impedance on the outputs. While output disable is selected, Q8–Q1 are placed in the high-impedance state.

G.3 Protection and Verification

This section describes the code protection feature included in the EPROM cell, which protects code against copyright violations. Table G–2 shows the programming levels required for protecting the EPROM and verifying the protection. The paragraphs following the table describe the protect and verify functions.

Table G–2. TMS320E1x/P1x EPROM Protect and Protect Verify Mode Levels

Signal Name	'E15/P15'E17/P17 DIP Pin	'E14/P14 Pin	TMS27C64 Pin	EPROM Protect	Protect Verify
$\overline{E}$	25	19	20	V_{IH}	V_{IL}
$\overline{G}$	24	23	22	V_{IH}	V_{IL}
$\overline{PGM}$	23	16	27	V_{IH}	V_{IH}
V_{PP}	3	18	1	V_{PP}	V_{CCP}
V_{CC}	30	4, 33	28	V_{CCP}	V_{CCP}
V_{SS}	10	3, 34	14	V_{SS}	V_{SS}
CLKIN	8	24	14	V_{SS}	V_{SS}
$\overline{RS}$	4	–	14	V_{SS}	V_{SS}
EPT	5	17	26	V_{PP}	V_{PP}
Q8–Q1	18–11	29–32,37,38,41,42	19–15,13–11	$Q_8=\overline{PULSE}$	Q_8=RBIT
A12–A10	26–28	15,11,10	2,23,21	X	X
A9–A7	29,34,35	8,7,2	24,25,3	X	X
A6	36	1	4	X	V_{IL}
A5	37	68	5	X	X
A4	38	67	6	V_{IH}	X
A3–A0	39,40,1,2	66,65,56,55	7–10	X	X

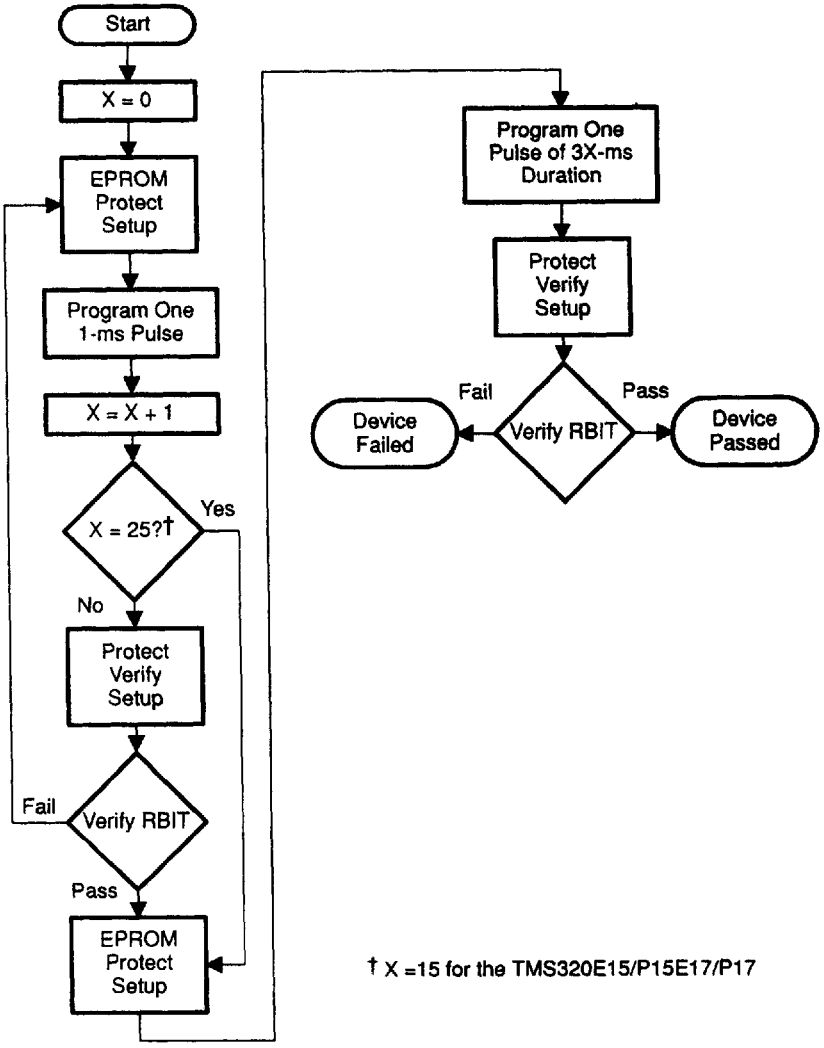
LEGEND:
V_{IH} = TTL high level; V_{IL} = low-level TTL, V_{CC} = 5 ± 0.25 V;
V_{PP} = 12.5 ± 0.5 V (FAST);
V_{CCP} = 6 ± 0.25 V (FAST) or 6.5 ± 0.25 V (SNAP!, TMS320E14/P14 only);
V_{PP} = 13 ± 0.25V (SNAP!, TMS320E14/P14 only)
X = don't care; PULSE = low-going TTL level pulse; RBIT = ROM protect bit

G.3.1 EPROM Protection

The EPROM protection facility is used to completely disable reading of the EPROM contents to guarantee security of proprietary algorithms. This facility is implemented through a unique EPROM cell called the RBIT (ROM protect bit) cell. Once the contents to be protected are programmed into the EPROM, the RBIT is programmed, disabling access to the EPROM contents and disabling the microprocessor mode on the device. Once programmed, the RBIT can only be cleared by erasing the entire EPROM array with ultraviolet light, thereby maintaining security of the proprietary algorithm. Programming the RBIT is

accomplished using the EPROM protection cycle, which consists of setting the $\overline{E}$, $\overline{G}$, $\overline{PGM}$, and A4 pins high, V_{PP} and EPT to 12.5 ± 0.5 V, and pulsing Q8 low. The complete sequence of operations involved in programming the RBIT is shown in the flowchart of Figure G–10. The required setups in the figure are detailed in Table G–2.

Figure G–10. EPROM Protection Flowchart



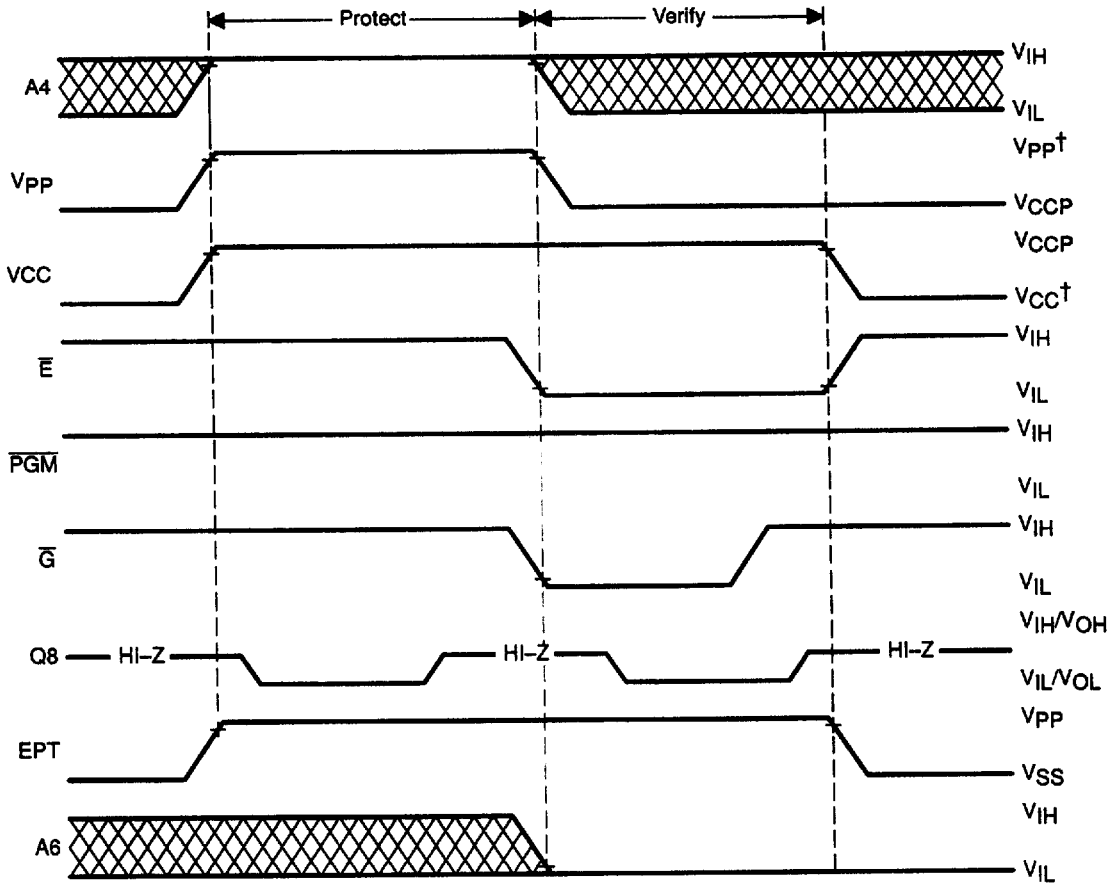
† X =15 for the TMS320E15/P15E17/P17

G.3.2 Protect Verify

Protect verify is used following the EPROM protection to verify correct programming of the RBIT (see Figure G–10). When using protect verify, Q8 outputs the state of the RBIT. When RBIT = 1, the EPROM is unprotected;

when RBIT = 0, the EPROM is protected. The EPROM protection and verify timings are shown in Figure G–11.

Figure G–11. EPROM Protection Timing



† 12.5-V V_{PP} and 6-V V_{CC} for FAST programming;
13-V V_{PP} and 6.5-V V_{CC} for SNAP! pulse programming (TMS320E14/P14 only).

Index

A

B

C

L

M

N

O

P

Q

R

S

T

V

W

X

Z