

Qualcomm uC extensions

Version 0.10.0, 2025-04-16

Table of Contents

Licensing and Acknowledgements	1
Copyright and license information	2
Acknowledgements	3
Versions	4
Version History	4
0.1.0	4
0.2.0	4
0.3.0	4
0.4.0	5
0.5.0	5
0.6.0	6
0.7.0	7
0.8.0	9
0.9.0	9
0.10.0	10
1. Conventions	12
2. Extension description	13
2.1. Sub-extensions	15
2.1.1. Xqcia (0.6.0)	15
2.1.2. Xqciac (0.3.0)	15
2.1.3. Xqcibi (0.2.0)	15
2.1.4. Xqcibm (0.7.0)	15
2.1.5. Xqcicli (0.3.0)	16
2.1.6. Xqcicm (0.2.0)	16
2.1.7. Xqcics (0.2.0)	16
2.1.8. Xqcicsr (0.3.0)	16
2.1.9. Xqciint (0.7.0)	16
2.1.10. Xqciio (0.1.0)	16
2.1.11. Xqcilb (0.2.0)	16
2.1.12. Xqcili (0.2.0)	16
2.1.13. Xqcilia (0.2.0)	17
2.1.14. Xqcilo (0.3.0)	17
2.1.15. Xqcilsm (0.5.0)	17
2.1.16. Xqcisim (0.2.0)	17
2.1.17. Xqcisls (0.2.0)	17
2.1.18. Xqcisync (0.2.0)	17
3. Instruction summary	18
3.1. Instructions by sub-extension	23
4. CSR summary	30
5. Csrs (in alphabetical order)	32
5.1. qc.mcause	33

5.1.1. Attributes	33
5.1.2. Format	33
5.1.3. Field Summary	33
5.1.4. Fields	34
INT	34
NMIE	34
MPDT	34
MNPIE	34
MPIE	35
MIE	35
EXCP	36
NMIP	36
MNPIL	36
MPIL	37
MIL	37
EXCCODE	37
5.2. qc.mclcie0	38
5.2.1. Attributes	38
5.2.2. Format	38
5.2.3. Field Summary	38
5.2.4. Fields	39
IRQ0	39
IRQ1	39
IRQ2	40
IRQ3	40
IRQ4	40
IRQ5	41
IRQ6	41
IRQ7	41
IRQ8	42
IRQ9	42
IRQ10	42
IRQ11	43
IRQ12	43
IRQ13	43
IRQ14	44
IRQ15	44
IRQ16	44
IRQ17	45
IRQ18	45
IRQ19	45
IRQ20	46
IRQ21	46

IRQ22	46
IRQ23	47
IRQ24	47
IRQ25	47
IRQ26	48
IRQ27	48
IRQ28	48
IRQ29	49
IRQ30	49
IRQ31	49
5.3. qc.mclcie1	50
5.3.1. Attributes	50
5.3.2. Format	50
5.3.3. Field Summary	50
5.3.4. Fields	51
IRQ32	51
IRQ33	51
IRQ34	52
IRQ35	52
IRQ36	52
IRQ37	53
IRQ38	53
IRQ39	53
IRQ40	54
IRQ41	54
IRQ42	54
IRQ43	55
IRQ44	55
IRQ45	55
IRQ46	56
IRQ47	56
IRQ48	56
IRQ49	57
IRQ50	57
IRQ51	57
IRQ52	58
IRQ53	58
IRQ54	58
IRQ55	59
IRQ56	59
IRQ57	59
IRQ58	60
IRQ59	60

IRQ60	60
IRQ61	61
IRQ62	61
IRQ63	61
5.4. qc.mclcie2	62
5.4.1. Attributes	62
5.4.2. Format	62
5.4.3. Field Summary	62
5.4.4. Fields	63
IRQ64	63
IRQ65	63
IRQ66	64
IRQ67	64
IRQ68	64
IRQ69	65
IRQ70	65
IRQ71	65
IRQ72	66
IRQ73	66
IRQ74	66
IRQ75	67
IRQ76	67
IRQ77	67
IRQ78	68
IRQ79	68
IRQ80	68
IRQ81	69
IRQ82	69
IRQ83	69
IRQ84	70
IRQ85	70
IRQ86	70
IRQ87	71
IRQ88	71
IRQ89	71
IRQ90	72
IRQ91	72
IRQ92	72
IRQ93	73
IRQ94	73
IRQ95	73
5.5. qc.mclcie3	74
5.5.1. Attributes	74

5.5.2. Format	74
5.5.3. Field Summary	74
5.5.4. Fields	75
IRQ96	75
IRQ97	75
IRQ98	76
IRQ99	76
IRQ100	76
IRQ101	77
IRQ102	77
IRQ103	77
IRQ104	78
IRQ105	78
IRQ106	78
IRQ107	79
IRQ108	79
IRQ109	79
IRQ110	80
IRQ111	80
IRQ112	80
IRQ113	81
IRQ114	81
IRQ115	81
IRQ116	82
IRQ117	82
IRQ118	82
IRQ119	83
IRQ120	83
IRQ121	83
IRQ122	84
IRQ123	84
IRQ124	84
IRQ125	85
IRQ126	85
IRQ127	85
5.6. qc.mclcie4	86
5.6.1. Attributes	86
5.6.2. Format	86
5.6.3. Field Summary	86
5.6.4. Fields	87
IRQ128	87
IRQ129	87
IRQ130	88

IRQ131	88
IRQ132	88
IRQ133	89
IRQ134	89
IRQ135	89
IRQ136	90
IRQ137	90
IRQ138	90
IRQ139	91
IRQ140	91
IRQ141	91
IRQ142.....	92
IRQ143.....	92
IRQ144.....	92
IRQ145	93
IRQ146	93
IRQ147	93
IRQ148	94
IRQ149	94
IRQ150	94
IRQ151	95
IRQ152	95
IRQ153	95
IRQ154	96
IRQ155	96
IRQ156	96
IRQ157.....	97
IRQ158.....	97
IRQ159.....	97
5.7. qc.mclcie5	98
5.7.1. Attributes	98
5.7.2. Format	98
5.7.3. Field Summary.....	98
5.7.4. Fields.....	99
IRQ160	99
IRQ161	99
IRQ162.....	100
IRQ163.....	100
IRQ164.....	100
IRQ165	101
IRQ166	101
IRQ167	101
IRQ168	102

IRQ169	102
IRQ170	102
IRQ171	103
IRQ172	103
IRQ173	103
IRQ174	104
IRQ175	104
IRQ176	104
IRQ177	105
IRQ178	105
IRQ179	105
IRQ180	106
IRQ181	106
IRQ182	106
IRQ183	107
IRQ184	107
IRQ185	107
IRQ186	108
IRQ187	108
IRQ188	108
IRQ189	109
IRQ190	109
IRQ191	109
5.8. qc.mclcie6	110
5.8.1. Attributes	110
5.8.2. Format	110
5.8.3. Field Summary	110
5.8.4. Fields	111
IRQ192	111
IRQ193	111
IRQ194	112
IRQ195	112
IRQ196	112
IRQ197	113
IRQ198	113
IRQ199	113
IRQ200	114
IRQ201	114
IRQ202	114
IRQ203	115
IRQ204	115
IRQ205	115
IRQ206	116

IRQ207	116
IRQ208	116
IRQ209	117
IRQ210	117
IRQ211	117
IRQ212	118
IRQ213	118
IRQ214	118
IRQ215	119
IRQ216	119
IRQ217	119
IRQ218	120
IRQ219	120
IRQ220	120
IRQ221	121
IRQ222	121
IRQ223	121
5.9. qc.mclcie7	122
5.9.1. Attributes	122
5.9.2. Format	122
5.9.3. Field Summary	122
5.9.4. Fields	123
IRQ224	123
IRQ225	123
IRQ226	124
IRQ227	124
IRQ228	124
IRQ229	125
IRQ230	125
IRQ231	125
IRQ232	126
IRQ233	126
IRQ234	126
IRQ235	127
IRQ236	127
IRQ237	127
IRQ238	128
IRQ239	128
IRQ240	128
IRQ241	129
IRQ242	129
IRQ243	129
IRQ244	130

IRQ245	130
IRQ246	130
IRQ247	131
IRQ248	131
IRQ249	131
IRQ250	132
IRQ251	132
IRQ252	132
IRQ253	133
IRQ254	133
IRQ255	133
5.10. qc.mclilvl00	134
5.10.1. Attributes	134
5.10.2. Format	134
5.10.3. Field Summary	134
5.10.4. Fields	134
IRQ0	134
IRQ1	135
IRQ2	135
IRQ3	135
IRQ4	136
IRQ5	136
IRQ6	136
IRQ7	137
5.11. qc.mclilvl01	138
5.11.1. Attributes	138
5.11.2. Format	138
5.11.3. Field Summary	138
5.11.4. Fields	138
IRQ8	138
IRQ9	139
IRQ10	139
IRQ11	139
IRQ12	140
IRQ13	140
IRQ14	140
IRQ15	141
5.12. qc.mclilvl02	142
5.12.1. Attributes	142
5.12.2. Format	142
5.12.3. Field Summary	142
5.12.4. Fields	142
IRQ16	142

IRQ17	143
IRQ18	143
IRQ19	143
IRQ20	144
IRQ21	144
IRQ22	144
IRQ23	145
5.13. qc.mclilvl03	146
5.13.1. Attributes	146
5.13.2. Format	146
5.13.3. Field Summary.....	146
5.13.4. Fields.....	146
IRQ24	146
IRQ25.....	147
IRQ26.....	147
IRQ27	147
IRQ28	148
IRQ29	148
IRQ30	148
IRQ31	149
5.14. qc.mclilvl04	150
5.14.1. Attributes	150
5.14.2. Format.....	150
5.14.3. Field Summary	150
5.14.4. Fields	150
IRQ32	150
IRQ33	151
IRQ34	151
IRQ35	151
IRQ36.....	152
IRQ37	152
IRQ38.....	152
IRQ39.....	153
5.15. qc.mclilvl05	154
5.15.1. Attributes	154
5.15.2. Format	154
5.15.3. Field Summary.....	154
5.15.4. Fields.....	154
IRQ40	154
IRQ41.....	155
IRQ42.....	155
IRQ43.....	155
IRQ44.....	156

IRQ45.....	156
IRQ46.....	156
IRQ47	157
5.16. qc.mclilvl06	158
5.16.1. Attributes	158
5.16.2. Format	158
5.16.3. Field Summary.....	158
5.16.4. Fields.....	158
IRQ48	158
IRQ49.....	159
IRQ50.....	159
IRQ51.....	159
IRQ52	160
IRQ53	160
IRQ54	160
IRQ55	161
5.17. qc.mclilvl07	162
5.17.1. Attributes.....	162
5.17.2. Format.....	162
5.17.3. Field Summary	162
5.17.4. Fields	162
IRQ56.....	162
IRQ57.....	163
IRQ58.....	163
IRQ59.....	163
IRQ60	164
IRQ61	164
IRQ62	164
IRQ63.....	165
5.18. qc.mclilvl08	166
5.18.1. Attributes	166
5.18.2. Format	166
5.18.3. Field Summary.....	166
5.18.4. Fields.....	166
IRQ64	166
IRQ65.....	167
IRQ66.....	167
IRQ67.....	167
IRQ68	168
IRQ69	168
IRQ70	168
IRQ71	169
5.19. qc.mclilvl09	170

5.19.1. Attributes	170
5.19.2. Format	170
5.19.3. Field Summary	170
5.19.4. Fields	170
IRQ72	170
IRQ73	171
IRQ74	171
IRQ75	171
IRQ76	172
IRQ77	172
IRQ78	172
IRQ79	173
5.20. qc.mclivil10	174
5.20.1. Attributes	174
5.20.2. Format	174
5.20.3. Field Summary	174
5.20.4. Fields	174
IRQ80	174
IRQ81	175
IRQ82	175
IRQ83	175
IRQ84	176
IRQ85	176
IRQ86	176
IRQ87	177
5.21. qc.mclivil11	178
5.21.1. Attributes	178
5.21.2. Format	178
5.21.3. Field Summary	178
5.21.4. Fields	178
IRQ88	178
IRQ89	179
IRQ90	179
IRQ91	179
IRQ92	180
IRQ93	180
IRQ94	180
IRQ95	181
5.22. qc.mclivil12	182
5.22.1. Attributes	182
5.22.2. Format	182
5.22.3. Field Summary	182
5.22.4. Fields	182

IRQ96	182
IRQ97	183
IRQ98	183
IRQ99	183
IRQ100	184
IRQ101	184
IRQ102	184
IRQ103	185
5.23. qc.mclivil13	186
5.23.1. Attributes	186
5.23.2. Format	186
5.23.3. Field Summary	186
5.23.4. Fields	186
IRQ104	186
IRQ105	187
IRQ106	187
IRQ107	187
IRQ108	188
IRQ109	188
IRQ110	188
IRQ111	189
5.24. qc.mclivil14	190
5.24.1. Attributes	190
5.24.2. Format	190
5.24.3. Field Summary	190
5.24.4. Fields	190
IRQ112	190
IRQ113	191
IRQ114	191
IRQ115	191
IRQ116	192
IRQ117	192
IRQ118	192
IRQ119	193
5.25. qc.mclivil15	194
5.25.1. Attributes	194
5.25.2. Format	194
5.25.3. Field Summary	194
5.25.4. Fields	194
IRQ120	194
IRQ121	195
IRQ122	195
IRQ123	195

IRQ124	196
IRQ125	196
IRQ126	196
IRQ127	197
5.26. qc.mclivil16	198
5.26.1. Attributes	198
5.26.2. Format	198
5.26.3. Field Summary	198
5.26.4. Fields	198
IRQ128	198
IRQ129	199
IRQ130	199
IRQ131	199
IRQ132	200
IRQ133	200
IRQ134	200
IRQ135	201
5.27. qc.mclivil17	202
5.27.1. Attributes	202
5.27.2. Format	202
5.27.3. Field Summary	202
5.27.4. Fields	202
IRQ136	202
IRQ137	203
IRQ138	203
IRQ139	203
IRQ140	204
IRQ141	204
IRQ142	204
IRQ143	205
5.28. qc.mclivil18	206
5.28.1. Attributes	206
5.28.2. Format	206
5.28.3. Field Summary	206
5.28.4. Fields	206
IRQ144	206
IRQ145	207
IRQ146	207
IRQ147	207
IRQ148	208
IRQ149	208
IRQ150	208
IRQ151	209

5.29. qc.mclivil19	210
5.29.1. Attributes.....	210
5.29.2. Format.....	210
5.29.3. Field Summary	210
5.29.4. Fields	210
IRQ152	210
IRQ153	211
IRQ154	211
IRQ155	211
IRQ156	212
IRQ157	212
IRQ158	212
IRQ159	213
5.30. qc.mclivil20.....	214
5.30.1. Attributes.....	214
5.30.2. Format	214
5.30.3. Field Summary	214
5.30.4. Fields	214
IRQ160	214
IRQ161	215
IRQ162	215
IRQ163	215
IRQ164	216
IRQ165	216
IRQ166	216
IRQ167	217
5.31. qc.mclivil21	218
5.31.1. Attributes	218
5.31.2. Format	218
5.31.3. Field Summary	218
5.31.4. Fields.....	218
IRQ168	218
IRQ169	219
IRQ170	219
IRQ171	219
IRQ172	220
IRQ173	220
IRQ174	220
IRQ175	221
5.32. qc.mclivil22	222
5.32.1. Attributes.....	222
5.32.2. Format	222
5.32.3. Field Summary	222

5.32.4. Fields	222
IRQ176	222
IRQ177	223
IRQ178	223
IRQ179	223
IRQ180	224
IRQ181	224
IRQ182	224
IRQ183	225
5.33. qc.mclilvl23	226
5.33.1. Attributes	226
5.33.2. Format	226
5.33.3. Field Summary	226
5.33.4. Fields	226
IRQ184	226
IRQ185	227
IRQ186	227
IRQ187	227
IRQ188	228
IRQ189	228
IRQ190	228
IRQ191	229
5.34. qc.mclilvl24	230
5.34.1. Attributes	230
5.34.2. Format	230
5.34.3. Field Summary	230
5.34.4. Fields	230
IRQ192	230
IRQ193	231
IRQ194	231
IRQ195	231
IRQ196	232
IRQ197	232
IRQ198	232
IRQ199	233
5.35. qc.mclilvl25	234
5.35.1. Attributes	234
5.35.2. Format	234
5.35.3. Field Summary	234
5.35.4. Fields	234
IRQ200	234
IRQ201	235
IRQ202	235

IRQ203	235
IRQ204	236
IRQ205	236
IRQ206	236
IRQ207	237
5.36. qc.mclcilvl26	238
5.36.1. Attributes	238
5.36.2. Format	238
5.36.3. Field Summary	238
5.36.4. Fields	238
IRQ208	238
IRQ209	239
IRQ210	239
IRQ211	239
IRQ212	240
IRQ213	240
IRQ214	240
IRQ215	241
5.37. qc.mclcilvl27	242
5.37.1. Attributes	242
5.37.2. Format	242
5.37.3. Field Summary	242
5.37.4. Fields	242
IRQ216	242
IRQ217	243
IRQ218	243
IRQ219	243
IRQ220	244
IRQ221	244
IRQ222	244
IRQ223	245
5.38. qc.mclcilvl28	246
5.38.1. Attributes	246
5.38.2. Format	246
5.38.3. Field Summary	246
5.38.4. Fields	246
IRQ224	246
IRQ225	247
IRQ226	247
IRQ227	247
IRQ228	248
IRQ229	248
IRQ230	248

IRQ231	249
5.39. qc.mclilvl29	250
5.39.1. Attributes	250
5.39.2. Format	250
5.39.3. Field Summary	250
5.39.4. Fields	250
IRQ232	250
IRQ233	251
IRQ234	251
IRQ235	251
IRQ236	252
IRQ237	252
IRQ238	252
IRQ239	253
5.40. qc.mclilvl30	254
5.40.1. Attributes	254
5.40.2. Format	254
5.40.3. Field Summary	254
5.40.4. Fields	254
IRQ240	254
IRQ241	255
IRQ242	255
IRQ243	255
IRQ244	256
IRQ245	256
IRQ246	256
IRQ247	257
5.41. qc.mclilvl31	258
5.41.1. Attributes	258
5.41.2. Format	258
5.41.3. Field Summary	258
5.41.4. Fields	258
IRQ248	258
IRQ249	259
IRQ250	259
IRQ251	259
IRQ252	260
IRQ253	260
IRQ254	260
IRQ255	261
5.42. qc.mclilvl32	262
5.42.1. Attributes	262
5.42.2. Format	262

5.42.3. Field Summary.....	262
5.42.4. Fields.....	263
IRQ0.....	263
IRQ1.....	263
IRQ2	264
IRQ3	264
IRQ4	264
IRQ5.....	265
IRQ6.....	265
IRQ7	265
IRQ8	266
IRQ9	266
IRQ10.....	266
IRQ11.....	267
IRQ12.....	267
IRQ13.....	267
IRQ14.....	268
IRQ15.....	268
IRQ16.....	268
IRQ17.....	269
IRQ18.....	269
IRQ19.....	269
IRQ20.....	270
IRQ21.....	270
IRQ22.....	270
IRQ23.....	271
IRQ24.....	271
IRQ25.....	271
IRQ26.....	272
IRQ27.....	272
IRQ28.....	272
IRQ29.....	273
IRQ30.....	273
IRQ31.....	273
5.43. qc.mclcip1	274
5.43.1. Attributes	274
5.43.2. Format	274
5.43.3. Field Summary.....	274
5.43.4. Fields.....	275
IRQ32	275
IRQ33.....	275
IRQ34.....	276
IRQ35.....	276

IRQ36	276
IRQ37	277
IRQ38	277
IRQ39	277
IRQ40	278
IRQ41	278
IRQ42	278
IRQ43	279
IRQ44	279
IRQ45	279
IRQ46	280
IRQ47	280
IRQ48	280
IRQ49	281
IRQ50	281
IRQ51	281
IRQ52	282
IRQ53	282
IRQ54	282
IRQ55	283
IRQ56	283
IRQ57	283
IRQ58	284
IRQ59	284
IRQ60	284
IRQ61	285
IRQ62	285
IRQ63	285
5.44. qc.mclcip2	286
5.44.1. Attributes	286
5.44.2. Format	286
5.44.3. Field Summary	286
5.44.4. Fields	287
IRQ64	287
IRQ65	287
IRQ66	288
IRQ67	288
IRQ68	288
IRQ69	289
IRQ70	289
IRQ71	289
IRQ72	290
IRQ73	290

IRQ74	290
IRQ75	291
IRQ76	291
IRQ77	291
IRQ78	292
IRQ79	292
IRQ80	292
IRQ81	293
IRQ82	293
IRQ83	293
IRQ84	294
IRQ85	294
IRQ86	294
IRQ87	295
IRQ88	295
IRQ89	295
IRQ90	296
IRQ91	296
IRQ92	296
IRQ93	297
IRQ94	297
IRQ95	297
5.45. qc.mclcip3	298
5.45.1. Attributes	298
5.45.2. Format	298
5.45.3. Field Summary	298
5.45.4. Fields	299
IRQ96	299
IRQ97	299
IRQ98	300
IRQ99	300
IRQ100	300
IRQ101	301
IRQ102	301
IRQ103	301
IRQ104	302
IRQ105	302
IRQ106	302
IRQ107	303
IRQ108	303
IRQ109	303
IRQ110	304
IRQ111	304

IRQ112	304
IRQ113	305
IRQ114	305
IRQ115	305
IRQ116	306
IRQ117	306
IRQ118	306
IRQ119	307
IRQ120	307
IRQ121	307
IRQ122	308
IRQ123	308
IRQ124	308
IRQ125	309
IRQ126	309
IRQ127	309
5.46. qc.mclicip4	310
5.46.1. Attributes	310
5.46.2. Format	310
5.46.3. Field Summary	310
5.46.4. Fields	311
IRQ128	311
IRQ129	311
IRQ130	312
IRQ131	312
IRQ132	312
IRQ133	313
IRQ134	313
IRQ135	313
IRQ136	314
IRQ137	314
IRQ138	314
IRQ139	315
IRQ140	315
IRQ141	315
IRQ142	316
IRQ143	316
IRQ144	316
IRQ145	317
IRQ146	317
IRQ147	317
IRQ148	318
IRQ149	318

IRQ150	318
IRQ151	319
IRQ152	319
IRQ153	319
IRQ154	320
IRQ155	320
IRQ156	320
IRQ157	321
IRQ158	321
IRQ159	321
5.47. qc.mclcip5	322
5.47.1. Attributes.....	322
5.47.2. Format	322
5.47.3. Field Summary	322
5.47.4. Fields.....	323
IRQ160.....	323
IRQ161.....	323
IRQ162	324
IRQ163	324
IRQ164	324
IRQ165	325
IRQ166	325
IRQ167	325
IRQ168	326
IRQ169	326
IRQ170	326
IRQ171.....	327
IRQ172.....	327
IRQ173.....	327
IRQ174	328
IRQ175	328
IRQ176	328
IRQ177	329
IRQ178	329
IRQ179	329
IRQ180	330
IRQ181	330
IRQ182	330
IRQ183	331
IRQ184	331
IRQ185	331
IRQ186.....	332
IRQ187.....	332

IRQ188	332
IRQ189	333
IRQ190	333
IRQ191	333
5.48. qc.mclicip6	334
5.48.1. Attributes	334
5.48.2. Format	334
5.48.3. Field Summary	334
5.48.4. Fields	335
IRQ192	335
IRQ193	335
IRQ194	336
IRQ195	336
IRQ196	336
IRQ197	337
IRQ198	337
IRQ199	337
IRQ200	338
IRQ201	338
IRQ202	338
IRQ203	339
IRQ204	339
IRQ205	339
IRQ206	340
IRQ207	340
IRQ208	340
IRQ209	341
IRQ210	341
IRQ211	341
IRQ212	342
IRQ213	342
IRQ214	342
IRQ215	343
IRQ216	343
IRQ217	343
IRQ218	344
IRQ219	344
IRQ220	344
IRQ221	345
IRQ222	345
IRQ223	345
5.49. qc.mclicip7	346
5.49.1. Attributes	346

5.49.2. Format.....	346
5.49.3. Field Summary	346
5.49.4. Fields	347
IRQ224.....	347
IRQ225.....	347
IRQ226	348
IRQ227	348
IRQ228	348
IRQ229	349
IRQ230	349
IRQ231	349
IRQ232	350
IRQ233	350
IRQ234	350
IRQ235	351
IRQ236	351
IRQ237	351
IRQ238	352
IRQ239	352
IRQ240	352
IRQ241	353
IRQ242	353
IRQ243	353
IRQ244	354
IRQ245	354
IRQ246	354
IRQ247	355
IRQ248	355
IRQ249	355
IRQ250	356
IRQ251	356
IRQ252	356
IRQ253.....	357
IRQ254.....	357
IRQ255.....	357
5.50. qc.mmcr.....	358
5.50.1. Attributes.....	358
5.50.2. Format.....	358
5.50.3. Field Summary	358
5.50.4. Fields	358
WP_EXEC.....	358
WP_WRITE	359
WP_READ.....	359

IRQNE	359
5.51. qc.mntvec	360
5.51.1. Attributes	360
5.51.2. Format	360
5.51.3. Field Summary	360
5.51.4. Fields	360
BASE	360
5.51.5. Software write	360
5.52. qc.mstkbottomaddr	362
5.52.1. Attributes	362
5.52.2. Format	362
5.52.3. Field Summary	362
5.52.4. Fields	362
STKADDR	362
RESERVED	363
5.53. qc.mstktopaddr	364
5.53.1. Attributes	364
5.53.2. Format	364
5.53.3. Field Summary	364
5.53.4. Fields	364
STKADDR	364
RESERVED	365
5.54. qc.mthreadptr	366
5.54.1. Attributes	366
5.54.2. Format	366
5.54.3. Field Summary	366
5.54.4. Fields	366
THREADPTR	366
5.55. qc.mwpendaddr0	367
5.55.1. Attributes	367
5.55.2. Format	367
5.55.3. Field Summary	367
5.55.4. Fields	367
ADDR	367
5.56. qc.mwpendaddr1	368
5.56.1. Attributes	368
5.56.2. Format	368
5.56.3. Field Summary	368
5.56.4. Fields	368
ADDR	368
5.57. qc.mwpendaddr2	369
5.57.1. Attributes	369
5.57.2. Format	369

5.57.3. Field Summary.....	369
5.57.4. Fields.....	369
ADDR	369
5.58. qc.mwpendaddr3.....	370
5.58.1. Attributes	370
5.58.2. Format	370
5.58.3. Field Summary	370
5.58.4. Fields	370
ADDR.....	370
5.59. qc.mwpstartaddr0	371
5.59.1. Attributes	371
5.59.2. Format.....	371
5.59.3. Field Summary	371
5.59.4. Fields	371
ADDR	371
5.60. qc.mwpstartaddr1	372
5.60.1. Attributes	372
5.60.2. Format	372
5.60.3. Field Summary.....	372
5.60.4. Fields.....	372
ADDR	372
5.61. qc.mwpstartaddr2	373
5.61.1. Attributes	373
5.61.2. Format.....	373
5.61.3. Field Summary	373
5.61.4. Fields	373
ADDR.....	373
5.62. qc.mwpstartaddr3.....	374
5.62.1. Attributes	374
5.62.2. Format	374
5.62.3. Field Summary.....	374
5.62.4. Fields.....	374
ADDR.....	374
6. Instructions (in alphabetical order)	375
6.1. qc.addsat	375
6.2. qc.addusat	377
6.3. qc.beqi	378
6.4. qc.bgei	379
6.5. qc.bgeui	380
6.6. qc.blti.....	381
6.7. qc.bltoi.....	382
6.8. qc.bnei	383
6.9. qc.brev32	384

6.10. qc.c.bexti	385
6.11. qc.c.bseti	386
6.12. qc.c.clrint	387
6.13. qc.c.delay	388
6.14. qc.c.di	389
6.15. qc.c.dir	390
6.16. qc.c.ei	391
6.17. qc.c.eir	392
6.18. qc.c.extu	393
6.19. qc.c.mienter.nest	394
6.20. qc.c.mienter	396
6.21. qc.c.mileaveret	398
6.22. qc.c.mnret	401
6.23. qc.c.mret	403
6.24. qc.c.muliadd	405
6.25. qc.c.mveqz	406
6.26. qc.c.ptrace	407
6.27. qc.c.setint	408
6.28. qc.c.sync	409
6.29. qc.c.syncr	410
6.30. qc.c.syncwf	411
6.31. qc.c.syncwl	412
6.32. qc.clo	413
6.33. qc.clrinti	414
6.34. qc.compress2	415
6.35. qc.compress3	416
6.36. qc.csrrwr	417
6.37. qc.csrrwri	418
6.38. qc.cto	419
6.39. qc.e.addai	420
6.40. qc.e.addi	421
6.41. qc.e.andai	422
6.42. qc.e.andi	423
6.43. qc.e.beqi	424
6.44. qc.e.bgei	425
6.45. qc.e.bgeui	426
6.46. qc.e.blتي	427
6.47. qc.e.blتي	428
6.48. qc.e.bnei	429
6.49. qc.e.j	430
6.50. qc.e.jal	431
6.51. qc.e.lb	432
6.52. qc.e.lbu	433

6.53. qc.e.lh	434
6.54. qc.e.lhu.....	435
6.55. qc.e.li	436
6.56. qc.e.lw	437
6.57. qc.e.orai	438
6.58. qc.e.ori	439
6.59. qc.e.sb	440
6.60. qc.e.sh	441
6.61. qc.e.sw	442
6.62. qc.e.xorai.....	443
6.63. qc.e.xori	444
6.64. qc.expand2	445
6.65. qc.expand3	446
6.66. qc.ext	447
6.67. qc.extd	448
6.68. qc.extdpr.....	449
6.69. qc.extdprh.....	450
6.70. qc.extdr	451
6.71. qc.extdu	452
6.72. qc.extdupr	453
6.73. qc.extduprh	454
6.74. qc.extdur	455
6.75. qc.extu.....	456
6.76. qc.insb	457
6.77. qc.insbh.....	458
6.78. qc.insbhr	459
6.79. qc.insbi	460
6.80. qc.insbpr	461
6.81. qc.insbprh.....	462
6.82. qc.insbr	463
6.83. qc.insbri.....	464
6.84. qc.inw	465
6.85. qc.li	466
6.86. qc.lieq	467
6.87. qc.lieqi	468
6.88. qc.lige	469
6.89. qc.ligei	470
6.90. qc.ligeu	471
6.91. qc.ligeui	472
6.92. qc.lilt.....	473
6.93. qc.lilti	474
6.94. qc.liltu	475
6.95. qc.liltui	476

6.96. qc.line.....	477
6.97. qc.linei.....	478
6.98. qc.lrb.....	479
6.99. qc.lrbu.....	480
6.100. qc.lrh.....	481
6.101. qc.lrhu.....	482
6.102. qc.lrw.....	483
6.103. qc.lwm.....	484
6.104. qc.lwmi.....	485
6.105. qc.muliadd.....	486
6.106. qc.mveq.....	487
6.107. qc.mveqi.....	488
6.108. qc.mvge.....	489
6.109. qc.mvgei.....	490
6.110. qc.mvgeu.....	491
6.111. qc.mvgeui.....	492
6.112. qc.mvlt.....	493
6.113. qc.mvlti.....	494
6.114. qc.mvltu.....	495
6.115. qc.mvltui.....	496
6.116. qc.mvne.....	497
6.117. qc.mvnei.....	498
6.118. qc.norm.....	499
6.119. qc.normeu.....	500
6.120. qc.normu.....	501
6.121. qc.outw.....	502
6.122. qc.pcoredump.....	503
6.123. qc.pexit.....	504
6.124. qc.ppreg.....	505
6.125. qc.ppregs.....	506
6.126. qc.pputc.....	507
6.127. qc.pputc_i.....	508
6.128. qc.pputs.....	509
6.129. qc.psyscall.....	510
6.130. qc.psyscall_i.....	511
6.131. qc.selecteqi.....	512
6.132. qc.selectieq.....	513
6.133. qc.selectieqi.....	514
6.134. qc.selectiieq.....	515
6.135. qc.selectiine.....	516
6.136. qc.selectine.....	517
6.137. qc.selectinei.....	518
6.138. qc.selectnei.....	519

6.139. qc.setinti.....	520
6.140. qc.setwm	521
6.141. qc.setwmi	522
6.142. qc.shladd	523
6.143. qc.shlsat	524
6.144. qc.shlusat	525
6.145. qc.srb.....	526
6.146. qc.srh	527
6.147. qc.srw.....	528
6.148. qc.subsat	529
6.149. qc.subusat	530
6.150. qc.swm	531
6.151. qc.swmi	532
6.152. qc.sync	533
6.153. qc.syncr	534
6.154. qc.syncwf	535
6.155. qc.syncwl	536
6.156. qc.wrap	537
6.157. qc.wrapi	538
7. IDL Functions.....	539
7.1. abort_current_instruction (builtin)	539
7.2. access_check.....	539
7.3. assert (builtin)	539
7.4. cached_translation (generated)	539
7.5. creg2reg.....	540
7.6. current_translation_mode.....	540
7.7. delay (builtin).....	543
7.8. effective_ldst_mode.....	543
7.9. exception_handling_mode	543
7.10. fence_tso (builtin)	544
7.11. get_and_validate_stack_pointer	544
7.12. highest_set_bit	545
7.13. ialign	545
7.14. implemented? (generated)	545
7.15. implemented_version? (generated).....	545
7.16. in_naturally_aligned_region?	546
7.17. is_naturally_aligned	546
7.18. iss_syscall (builtin)	546
7.19. jump_halfword.....	546
7.20. lowest_set_bit	547
7.21. maybe_cache_translation (generated)	547
7.22. misaligned_is_atomic?	547
7.23. mode	548

7.24. mtval_for.....	548
7.25. mtval_readonly?.....	549
7.26. notify_mode_change (builtin).....	549
7.27. pma_applies? (builtin)	550
7.28. raise	550
7.29. raise_precise	550
7.30. read_device_32 (builtin).....	552
7.31. read_memory.....	552
7.32. read_memory_aligned	553
7.33. read_physical_memory	554
7.34. read_physical_memory_16 (builtin)	554
7.35. read_physical_memory_32 (builtin).....	554
7.36. read_physical_memory_8 (builtin).....	554
7.37. refresh_pending_interrupts.....	554
7.38. set_mode.....	555
7.39. sext.....	556
7.40. sync_read_after_write_device (builtin)	556
7.41. sync_write_after_read_device (builtin)	556
7.42. translate	556
7.43. unpredictable (builtin).....	558
7.44. write_device_32 (builtin).....	558
7.45. write_memory	558
7.46. write_memory_aligned.....	559
7.47. write_physical_memory	559
7.48. write_physical_memory_16 (builtin)	560
7.49. write_physical_memory_32 (builtin).....	560
7.50. write_physical_memory_8 (builtin).....	560
7.51. xlen	560

Licensing and Acknowledgements



This document is in the [Frozen state](#).

Change is extremely unlikely. A high threshold will be used, and a change will only occur because of some truly critical issue being identified during the public review cycle. Any other desired or needed changes can be the subject of a follow-on new extension.

Copyright and license information

This document is released under the [Creative Commons Attribution 4.0 International License](#).

Copyright 2025 by Qualcomm Technologies, Inc..

Acknowledgements

Contributors to version 0.10.0 of the specification (in alphabetical order) include:

- Derek Hower <dhower@qti.qualcomm.com> (Qualcomm Technologies, Inc.)
- Albert Yosher <ayosher@qti.qualcomm.com> (Qualcomm Technologies, Inc.)

We express our gratitude to everyone that contributed to, reviewed or improved this specification through their comments and questions.

Versions

This specification documents version 0.10.0 of Xqci.

Version History

0.1.0

State

development

0.2.0

State

development

Changes

- Splits Xqci into sub-extensions based on instruction type
- Removed shXadd instructions in favor of generic shladd

0.3.0

State

development

Changes

- Rename extension and sub extensions to match toolchain guidelines
- Rename mnemonics to match toolchain guidelines
- Ensure every instruction belongs to at least one sub extension

Implies

- Xqcia (0.1.0)
- Xqciac (0.1.0)
- Xqcibi (0.1.0)
- Xqcibm (0.1.0)
- Xqicli (0.1.0)
- Xqicm (0.1.0)
- Xqcics (0.1.0)
- Xqcicsr (0.1.0)
- Xqciint (0.1.0)
- Xqcilb (0.1.0)
- Xqcili (0.1.0)
- Xqcilia (0.1.0)

- Xqcilo (0.1.0)
- Xqcilsm (0.1.0)
- Xqcisls (0.1.0)

0.4.0

State

frozen

Changes

- Add information about instruction formats of each instruction
- Fix description and functionality of qc.c.extu instruction
- Fix description and functionality of qc.shladd instruction

Implies

- Xqcia (0.2.0)
- Xqciac (0.2.0)
- Xqcibi (0.2.0)
- Xqcibm (0.2.0)
- Xqcicli (0.2.0)
- Xqcicm (0.2.0)
- Xqcics (0.2.0)
- Xqcicsr (0.2.0)
- Xqciint (0.2.0)
- Xqcilb (0.2.0)
- Xqcili (0.2.0)
- Xqcilia (0.2.0)
- Xqcilo (0.2.0)
- Xqcilsm (0.2.0)
- Xqcisls (0.2.0)

Requires

- Zca, version $\geq 1.0.0$

0.5.0

State

frozen

Changes

- Added Xqciio sub-extension
- Added Xqcisim sub-extension

- Added Xqcisync sub-extension
- Rename instruction of qc.muladdi to qc.muliadd
- Rename instruction of qc.c.muladdi to qc.c.muliadd
- Fix description of qc.shladd instruction
- Fix description and functionality of qc.c.extu instruction
- Fix description and functionality of qc.wrapi instruction
- Fix description of qc.swmi, qc.lwmi and qc.setwmi instructions
- Fix description of qc.mclici* CSRs to reflect being part of Xqciint custom extension
- Fix description of qc.setinti and qc.clrinti instructions
- Fix to make qc.c.mret and qc.c.mnret visible

Implies

- Xqcia (0.3.0)
- Xqciac (0.3.0)
- Xqcibi (0.2.0)
- Xqcibm (0.3.0)
- Xqicli (0.2.0)
- Xqcicm (0.2.0)
- Xqcics (0.2.0)
- Xqcicsr (0.2.0)
- Xqciint (0.3.0)
- Xqciio (0.1.0)
- Xqcilb (0.2.0)
- Xqcili (0.2.0)
- Xqcilia (0.2.0)
- Xqcilo (0.2.0)
- Xqcilsm (0.3.0)
- Xqcisim (0.1.0)
- Xqcisls (0.2.0)
- Xqcisync (0.1.0)

Requires

- Zca, version $\geq 1.0.0$

0.6.0

State

frozen

Changes

- Fix encoding of qc.c.extu instruction
- Fix encoding of qc.swmi instruction
- Fix decoding of qc.pputci instruction
- Fix decoding of qc.c.delay instruction (state that immediate cannot be 0)
- Rename qc.slasat → qc.shlsat
- Rename qc.sllsat → qc.shlusat
- Add requirement to include Zca extension for Xqcisim since it has 16-bit instructions
- Add requirement to include Zca extension for Xqcisync since it has 16-bit instructions
- Remove qc.flags CSR

Implies

- Xqcia (0.4.0)
- Xqciac (0.3.0)
- Xqcibi (0.2.0)
- Xqcibm (0.4.0)
- Xqicli (0.2.0)
- Xqicm (0.2.0)
- Xqcics (0.2.0)
- Xqcicsr (0.3.0)
- Xqciint (0.3.0)
- Xqciio (0.1.0)
- Xqcilb (0.2.0)
- Xqcili (0.2.0)
- Xqcilia (0.2.0)
- Xqcilo (0.2.0)
- Xqcilsm (0.4.0)
- Xqcisim (0.2.0)
- Xqcisls (0.2.0)
- Xqcisync (0.2.0)

Requires

- Zca, version >= 1.0.0

0.7.0

State

frozen

Changes

- Fix typos in description of qc.shlsat and qc.shlusat instructions

- Fix bug in qc.shlsat that caused wrong IDL code result
- Fix clobbering of saturation results in [add|addu|sub]sat instructions
- Rename all custom CSRs of the Xqciint extension to use dot as separator in the name
- Fix addresses of qc.mclicie* CSRs to 0x7f8-0x7ff (was copy & paste from qc.mclicip*)
- Add missing CSR registers qc.mclivil00 - qc.mclivil31
- Add missing CSR registers qc.mwpstartaddr0 - qc.mwpstartaddr3 and qc.mwpendaddr0 - qc.mwpendaddr3
- Add missing CSR registers qc.mstkbottomaddr and qc.mstktopaddr
- Add missing CSR registers qc.mmcr, qc.mhreadptr, qc.mcause
- Remove CSR registers qc.mncause, qc.mnepc
- Fix IDL code for qc.c.mienter, qc.c.mienter.nest, qc.c.mileaveret, qc.c.mret, qc.c.mnret
- Fix IDL code for qc.c.ei, qc.c.eir, qc.c.di, qc.c.dir
- Add list of supported custom exceptions
- Fix qc.cto instruction IDL code
- Change width calculations for qc.extdpr, qc.extdprh, qc.extdr, qc.extdupr, qc.extduprh, qc.extdur,
- Change width calculations for qc.insbhr, qc.insbpr, qc.insbprh, qc.insbr, qc.insbri

Implies

- Xqcia (0.5.0)
- Xqciac (0.3.0)
- Xqcibi (0.2.0)
- Xqcibm (0.5.0)
- Xqcicli (0.2.0)
- Xqcicm (0.2.0)
- Xqcics (0.2.0)
- Xqcicsr (0.3.0)
- Xqciint (0.4.0)
- Xqciio (0.1.0)
- Xqcilb (0.2.0)
- Xqcili (0.2.0)
- Xqcilia (0.2.0)
- Xqcilo (0.2.0)
- Xqcilsm (0.4.0)
- Xqcisim (0.2.0)
- Xqcisls (0.2.0)
- Xqcisync (0.2.0)

Requires

- Zca, version >= 1.0.0

0.8.0**State**

frozen

Changes

- Fix rs1 cannot be 31 for qc.extdu, qc.extd, qc.extdur, instructions
- Fix rs1 cannot be 31 for qc.extduprh, qc.extdprh, qc.extdupr, qc.extdr qc.extdpr instructions
- Fix typos in IDL code (missing ') ') for qc.extdpr, qc.extdr instructions
- Fix IDL code and description to look correct in PDF for qc.insbhr and qc.insbh instructions
- Fix Xqci extension description to reflect correct 48-bit format field names
- Fix IDL code to to match description for qc.insbr instruction
- Add stack checks to qc.c.mienter, qc.c.mienter.nest, qc.c.mileaveret

Implies

- Xqcia (0.5.0)
- Xqciac (0.3.0)
- Xqcibi (0.2.0)
- Xqcibm (0.6.0)
- Xqicli (0.2.0)
- Xqcicm (0.2.0)
- Xqcics (0.2.0)
- Xqcicsr (0.3.0)
- Xqciint (0.5.0)
- Xqciio (0.1.0)
- Xqcilb (0.2.0)
- Xqcili (0.2.0)
- Xqcilia (0.2.0)
- Xqcilo (0.2.0)
- Xqcilsm (0.4.0)
- Xqcisim (0.2.0)
- Xqcisls (0.2.0)
- Xqcisync (0.2.0)

Requires

- Zca, version $\geq$ 1.0.0

0.9.0**State**

frozen

Changes

- Fix IDL code sign extension logic for qc.e.lb and qc.e.lh instructions
- Fix IDL code sign extension logic for qc.ext and qc.extd instructions
- Fix IDL code sign extension logic for qc.extdpr, qc.extdprh and qc.extdr instructions
- Fix IDL code and description increasing shift to 6 bit for qc.extdr and qc.extdur instructions
- Fix IDL code and description increasing shift to 6 bit for qc.extdpr and qc.extdprh instructions
- Fix IDL code and description increasing shift to 6 bit for qc.extdupr and qc.extduprh instructions
- Fix IDL code sign extension logic for qc.lieq instruction
- Fix wrong mantissa bit selection in qc.norm, qc.normu and qc.normeu instructions
- Fix wrong exponent calculation in qc.normeu instruction
- Fix IDL code and description of qc.setwm instruction to state that number of words written 0..31.
- Fix IDL code for Smdbltrp and Smrnmi spec compatibility for qc.c.mret and qc.c.mnret instructions

Implies

- Xqcia (0.6.0)
- Xqciac (0.3.0)
- Xqcibi (0.2.0)
- Xqcibm (0.7.0)
- Xqicli (0.3.0)
- Xqicm (0.2.0)
- Xqcics (0.2.0)
- Xqcicsr (0.3.0)
- Xqciint (0.6.0)
- Xqciio (0.1.0)
- Xqcilb (0.2.0)
- Xqcili (0.2.0)
- Xqcilia (0.2.0)
- Xqcilo (0.3.0)
- Xqcilsm (0.5.0)
- Xqcisim (0.2.0)
- Xqcisls (0.2.0)
- Xqcisync (0.2.0)

Requires

- Zca, version $\geq$ 1.0.0

0.10.0

State

frozen

Changes

- Fix IDL code for Smdbltrp and Smrnmi spec compatibility for qc.c.mileaveret instruction

Implies

- Xqcia (0.6.0)
- Xqciac (0.3.0)
- Xqcibi (0.2.0)
- Xqcibm (0.7.0)
- Xqicli (0.3.0)
- Xqicm (0.2.0)
- Xqcics (0.2.0)
- Xqcicsr (0.3.0)
- Xqciint (0.7.0)
- Xqciio (0.1.0)
- Xqcilb (0.2.0)
- Xqcili (0.2.0)
- Xqcilia (0.2.0)
- Xqcilo (0.3.0)
- Xqcilsm (0.5.0)
- Xqcisim (0.2.0)
- Xqcisls (0.2.0)
- Xqcisync (0.2.0)

Requires

- Zca, version >= 1.0.0

Chapter 1. Conventions

Version requirements are specified as conditions using the following operators:

Operator	Meaning
~> VERSION	Accepts any version that is <i>compatible</i> with VERSION. By default, a version A is compatible with a version B if A's version number is greater than or equal to version B. If that default assumption is ever broken, it will be noted in the list of extension versions.
= VERSION	Accepts only version VERSION.
>= VERSION	Accepts any version greater than or equal to VERSION.
<= VERSION	Accepts any version less than or equal to VERSION.
> VERSION	Accepts any version greater than VERSION.
< VERSION	Accepts any version less than VERSION.

Chapter 2. Extension description

The Xqci extension includes a set of instructions that improve RISC-V code density and performance in microcontrollers. It fills several gaps:

Long immediates

This extension provides wide-immediate versions of loads, stores, branches, jumps, and several arithmetic operations.

Addressing modes

New indexed addressing modes are added for load and store instructions.

Load/store multiple

Xqci adds instructions that load multiple sequential values from memory into multiple registers. Analogous instructions for stores are also included.

Branch immediate instructions

The base RISC-V ISA provides conditional branches that compare two register values. Xqci adds conditional branch instructions that compare an immediate value to a register value, reducing a common code sequence into a single instruction.

Conditional instructions

Xqci includes a variety of conditional select instructions that pick one of two operand values based on the result of a condition and conditional move instructions that either move a value or retain a value in a destination register based on the result of a condition. Conditional select and conditional move instructions help reduce code size and avoid branches in common code sequences.

Extra bit manipulation instructions

Xqci expands upon the standard B extension by adding instructions for bit insertion/extraction, bit set/clear, sign and zero extension, and bit counting instructions.

Fast interrupt instructions

Xqci adds instructions to accelerate interrupt handling, including instructions to quickly enable/disable interrupts and instructions to automatically save an interrupt frame.

Saturating arithmetic

Xqci adds saturating arithmetic instructions to improve performance of fixed-point calculations.

Xqci adds 16, 32, and 48-bit instruction encodings. The 16 and 32-bit encodings are allocated in the custom opcode space so as not to conflict with standard RISC-V instructions. The 48-bit instructions follow the guidance for 48-bit instructions in the RISC-V standard, but are not allocated in reserved custom space since no such space has been defined by RISC-V International.

Extension-specific instruction formats

QC.EAI format used for 48-bit instructions that operate on 32-bit immediate argument.

Field	Start bit	Width	Description
opcode	0	7	Opcode field of 48-bit instructions is 0x1F
rd	7	5	Destination register
funct3	12	3	Function field identifying instruction group
f1	15	1	Secondary function field
imm[31:0]	16	32	Immediate operand of 32 bits

QC.EI format used for 48-bit instructions that operate on 26-bit immediate argument, including loads.

Field	Start bit	Width	Description
opcode	0	7	Opcode field of 48-bit instructions is 0x1F
rd	7	5	Destination register
funct3	12	3	Function field identifying instruction group
rs1	15	5	Register argument
imm[9:0]	20	10	Immediate operand of 26 bits, the 10 LSBs
funct2	30	2	Secondary function field
imm[25:10]	32	16	Immediate operand of 26 bits, the 16 MSBs

QC.EB format used for 48-bit branch instructions that compare register with 16-bit immediate.

Field	Start bit	Width	Description
opcode	0	7	Opcode field of 48-bit instructions is 0x1F
imm[4:1,11]	7	5	Bits of immediate value of branch target
funct3	12	3	Function field identifying instruction group
rs1	15	5	Register argument
funct5	20	5	Secondary function field
imm[12,10:5]	25	7	Bits of immediate value of branch target
simm[15:0]	32	16	Immediate operand of 16 bits to compare with register

QC.EJ format used for 48-bit jump/call instructions with 32-bit immediate target address.

Field	Start bit	Width	Description
opcode	0	7	Opcode field of 48-bit instructions is 0x1F
imm[4:1,11]	7	5	Bits of immediate value of branch target
funct3	12	3	Function field identifying instruction group
funct2	15	2	Secondary function field
imm[15:13]	17	3	Bits of immediate value of branch target
funct5	20	5	Secondary function field
imm[12,10:5]	25	7	Bits of immediate value of branch target

Field	Start bit	Width	Description
imm[31:16]	32	16	The 16 MSBs of immediate value of branch target

QC.ES format used for 48-bit store instructions with 26-bit immediate offset.

Field	Start bit	Width	Description
opcode	0	7	Opcode field of 48-bit instructions is 0x1F
imm[4:0]	7	5	Immediate operand of 26 bits offset, the 5 LSBs
funct3	12	3	Function field identifying instruction group
rs1	15	5	Register argument used as base address
rs2	20	5	Register argument to be saved
imm[9:5]	25	5	Immediate operand of 26 bits offset, the 5 bits
funct2	30	2	Secondary function field
imm[25:10]	32	16	Immediate operand of 26 bits offset, the 16 MSBs

2.1. Sub-extensions

Xqci defines the following 18 sub-extensions:

2.1.1. Xqcia (0.6.0)

The Xqcia extension includes eleven instructions to perform integer arithmetic.

2.1.2. Xqciac (0.3.0)

The Xqciac extension includes three instructions to accelerate common address calculations.

Xqciac requires:

- Zca, version $\geq 1.0.0$

2.1.3. Xqcibi (0.2.0)

The Xqcibi extension includes twelve conditional branch instructions that use an immediate operand for a source.

All instructions in Xqcibi following the same relocation type as standard RISC-V branches in the base architecture (e.g., BEQ, BNE, etc.).

Xqcibi requires:

- Zca, version $\geq 1.0.0$

2.1.4. Xqcibm (0.7.0)

The Xqcibm extension includes thirty eight instructions that perform bit manipulation, include

insertion and extraction.

Xqcibm requires:

- Zca, version $\geq 1.0.0$

2.1.5. Xqcicli (0.3.0)

The Xqcicli extension includes twelve instructions that conditionally load an immediate value.

2.1.6. Xqcicm (0.2.0)

The Xqcicm extension includes thirteen conditional move instructions.

Xqcicm requires:

- Zca, version $\geq 1.0.0$

2.1.7. Xqcics (0.2.0)

The Xqcics extension includes eight conditional select instructions.

2.1.8. Xqcicsr (0.3.0)

The Xqcicsr extension contains two instructions to read/write CSR which index is in register and not immediate.

2.1.9. Xqciint (0.7.0)

The Xqciint extension includes eleven instructions to accelerate interrupt servicing by performing common actions during ISR prologue/epilogue.

Xqciint requires:

- Zca, version $\geq 1.0.0$

2.1.10. Xqciio (0.1.0)

The Xqciio extension includes two instructions to access external non-memory-mapped devices for input and output

2.1.11. Xqcilb (0.2.0)

The Xqcilb extension includes two 48-bit instructions to encode a long branch.

Xqcilb requires:

- Zca, version $\geq 1.0.0$

2.1.12. Xqcili (0.2.0)

The Xqcili extension includes a two instructions that load large immediates than is available with the base RISC-V ISA.

Xqcili requires:

- Zca, version $\geq 1.0.0$

2.1.13. Xqcilia (0.2.0)

The Xqcilia extension includes eight 48-bit instructions that perform arithmetic using large immediates.

Xqcilia requires:

- Zca, version $\geq 1.0.0$

2.1.14. Xqcilo (0.3.0)

The Xqcilo extension includes eight 48-bit load/stores instructions that use an offset larger than can be found in the base RISC-V ISA.

2.1.15. Xqcilsm (0.5.0)

The Xqcilsm extension includes six instructions that transfer multiple values between registers and memory.

2.1.16. Xqcisim (0.2.0)

The Xqcisim extension includes ten hint instructions to interface simulation environment. On real target any instruction from this extension executed as "no-operation" and have no effect.

Xqcisim requires:

- Zca, version $\geq 1.0.0$

2.1.17. Xqcisls (0.2.0)

The Xqcisls extension includes five load and three store instructions with a scaled index addressing mode.

2.1.18. Xqcisync (0.2.0)

The Xqcisync extension includes nine instructions, eight for non-memory-mapped devices synchronization and delay instruction. Synchronization instructions are kind of IO fences that work with special devices synchronization signals.

Xqcisync requires:

- Zca, version $\geq 1.0.0$

Chapter 3. Instruction summary

The following 157 instructions are affected by this extension:

RV32	RV64	Mnemonic	Instruction
✓		qc.addsat rd, rs1, rs2	Saturating signed addition
✓		qc.addusat rd, rs1, rs2	Saturating unsigned addition
✓		qc.beqi rs1, imm, offset	Branch on equal (Immediate)
✓		qc.bgei rs1, imm, offset	Branch on greater than or equal (immediate)
✓		qc.bgeui rs1, imm, offset	Branch on greater than or equal unsigned (immediate)
✓		qc.blti rs1, imm, offset	Branch on less than (immediate)
✓		qc.bltui rs1, imm, offset	Branch on less than unsigned (immediate)
✓		qc.bnei rs1, imm, offset	Branch on not equal (immediate)
✓		qc.brev32 rd, rs1	Reverse bit order
✓		qc.c.bexti rd, shamt	Single-Bit extract (Immediate)
✓		qc.c.bseti rd, shamt	Single-Bit set (Immediate)
✓		qc.c.clrint rs1	Clear interrupt (Register)
✓		`qc.c.delay`	Delay execution for immediate amount of cycles
✓		`qc.c.di`	Disable interrupts
✓		qc.c.dir rd	Disable interrupts (Register)
✓		`qc.c.ei`	Enable interrupts
✓		qc.c.eir rs1	Restore interrupts (Register)
✓		qc.c.extu rd, width	Extract bits unsigned
✓		`qc.c.mienter.nest`	Machine mode interrupt enter
✓		`qc.c.mienter`	Machine mode interrupt enter
✓		`qc.c.mileaveret`	Machine mode interrupt exit
✓		`qc.c.mnret`	Machine NMI Return
✓		`qc.c.mret`	Machine Exception Return
✓		qc.c.muliadd rd, rs1, uimm	Multiply and accumulate (Immediate)
✓		qc.c.mveqz rd, rs1	Conditional Move if equal to zero
✓		`qc.c.ptrace`	Tracing pseudo-instruction (hint) working only in simulation environment
✓		qc.c.setint rs1	Set interrupt (Register)

✓	qc.c.sync slist	Non-memory-mapped device read-after-write synchronization to completion
✓	qc.c.syncr slist	Non-memory-mapped device read-after-write synchronization
✓	qc.c.syncwf slist	Non-memory-mapped device write-after-read synchronization
✓	qc.c.syncwl slist	Non-memory-mapped device write-after-read synchronization to completion
✓	qc.clo rd, rs1	Count leading ones
✓	qc.clrinti imm	Clear interrupt (Immediate)
✓	qc.compress2 rd, rs1	Bit compression (every 2nd bit)
✓	qc.compress3 rd, rs1	Bit compression (every 3rd bit)
✓	qc.csrrwr rd, rs1, rs2	Atomic Read/Write CSR (Register)
✓	qc.csrrwri rd, imm, rs2	Atomic Read/Write CSR (Register) Immediate
✓	qc.cto rd, rs1	Count trailing ones
✓	qc.e.addai rd, imm	Add immediate
✓	qc.e.addi rd, rs1, imm	Add immediate
✓	qc.e.andai rd, imm	And immediate
✓	qc.e.andi rd, rs1, imm	Add immediate
✓	qc.e.beqi rs1, imm, offset	Branch on equal (immediate)
✓	qc.e.bgei rs1, imm, offset	Branch on greater than or equal (immediate)
✓	qc.e.bgeui rs1, imm, offset	Branch on greater than or equal unsigned (immediate)
✓	qc.e.blti rs1, imm, offset	Branch on less than (immediate)
✓	qc.e.bltui rs1, imm, offset	Branch on less than unsigned (immediate)
✓	qc.e.bnei rs1, imm, offset	Branch on not equal (immediate)
✓	qc.e.j imm	Jump
✓	qc.e.jal imm	Jump and link
✓	qc.e.lb rd, imm(rs1)	Load byte
✓	qc.e.lbu rd, imm(rs1)	Load byte unsigned
✓	qc.e.lh rd, imm(rs1)	Load halfword
✓	qc.e.lhu rd, imm(rs1)	Load halfword unsigned
✓	qc.e.li rd, imm	Load immediate large
✓	qc.e.lw rd, imm(rs1)	Load word
✓	qc.e.orai rd, imm	Or immediate
✓	qc.e.ori rd, rs1, imm	Or immediate
✓	qc.e.sb rs2, imm(rs1)	Store byte

✓		qc.e.sh rs2, imm(rs1)	Store halfword
✓		qc.e.sw rs2, imm(rs1)	Store word
✓		qc.e.xorai rd, imm	Exclusive Or immediate
✓		qc.e.xori rd, rs1, imm	Exclusive Or immediate
✓		qc.expand2 rd, rs1	Bit expansion (every 2nd bit)
✓		qc.expand3 rd, rs1	Bit expansion (every 3rd bit)
✓		qc.ext rd, rs1, width, shamt	Extract bits signed
✓		qc.extd rd, rs1, width, shamt	Extract bits from pair signed (Immediate)
✓		qc.extdpr rd, rs1, rs2	Extract bits from pair signed, packed descriptor (Register)
✓		qc.extdprh rd, rs1, rs2	Extract bits from pair signed, packed descriptor high part (Register)
✓		qc.extdr rd, rs1, rs2	Extract bits from pair signed (Register)
✓		qc.extdu rd, rs1, width, shamt	Extract bits from pair unsigned (Immediate)
✓		qc.extdupr rd, rs1, rs2	Extract bits from pair unsigned, packed descriptor (Register)
✓		qc.extduprh rd, rs1, rs2	Extract bits from pair unsigned, packed descriptor high part (Register)
✓		qc.extdur rd, rs1, rs2	Extract bits from pair unsigned (Register)
✓		qc.extu rd, rs1, width, shamt	Extract bits unsigned
✓		qc.insb rd, rs1, width, shamt	Insert bits (Register)
✓		qc.insbh rd, rs1, width, shamt	Insert bits in 64-bit higher part (Register)
✓		qc.insbhr rd, rs1, rs2	Insert bits in 64-bit higher part (Register)
✓		qc.insbi rd, imm, width, shamt	Insert bits (Immediate)
✓		qc.insbpr rd, rs1, rs2	Insert bits, packed descriptor (Register)
✓		qc.insbprh rd, rs1, rs2	Insert bits, packed descriptor high part (Register)
✓		qc.insbr rd, rs1, rs2	Insert bits (Register)
✓		qc.insbri rd, rs1, imm	Insert bits (Immediate)
✓		qc.inw rd, imm(rs1)	Input word from non-memory-mapped device
✓		qc.li rd, imm	Load immediate large
✓		qc.lieq rd, rs1, rs2, simm	Conditional load immediate if equal (Register)
✓		qc.lieqi rd, rs1, imm, simm	Conditional load immediate if equal (Immediate)
✓		qc.lige rd, rs1, rs2, simm	Conditional load immediate if great or equal than (Register)
✓		qc.ligei rd, rs1, imm, simm	Conditional load immediate if great or equal than (Immediate)

✓		qc.ligeu rd, rs1, rs2, simm	Conditional load immediate if great or equal than unsigned (Register)
✓		qc.ligeui rd, rs1, imm, simm	Conditional load immediate if great or equal than unsigned (Immediate)
✓		qc.lilt rd, rs1, rs2, simm	Conditional load immediate if less than (Register)
✓		qc.lilti rd, rs1, imm, simm	Conditional load immediate if less than (Immediate)
✓		qc.liltu rd, rs1, rs2, simm	Conditional load immediate if less than unsigned (Register)
✓		qc.liltui rd, rs1, imm, simm	Conditional load immediate if less than unsigned (Immediate)
✓		qc.line rd, rs1, rs2, simm	Conditional load immediate if not equal (Register)
✓		qc.linei rd, rs1, imm, simm	Conditional load immediate if not equal (Immediate)
✓		qc.lrb rd, rs1, rs2, shamt	Load indexed byte
✓		qc.lrbu rd, rs1, rs2, shamt	Load indexed unsigned byte
✓		qc.lrh rd, rs1, rs2, shamt	Load indexed halfword
✓		qc.lrhu rd, rs1, rs2, shamt	Load indexed unsigned halfword
✓		qc.lrw rd, rs1, rs2, shamt	Load indexed word
✓		qc.lwm rd, rs2, imm(rs1)	Load word multiple
✓		qc.lwmi rd, length, imm(rs1)	Load word multiple (Immediate)
✓		qc.muliadd rd, rs1, imm	Multiply and accumulate (Immediate)
✓		qc.mveq rd, rs1, rs2, rs3	Conditional move if equal (Register)
✓		qc.mveqi rd, rs1, imm, rs3	Conditional move if equal (Immediate)
✓		qc.mvge rd, rs1, rs2, rs3	Conditional move if great or equal than (Register)
✓		qc.mvgei rd, rs1, imm, rs3	Conditional move if great or equal than (Immediate)
✓		qc.mvgeu rd, rs1, rs2, rs3	Conditional move if great or equal than unsigned (Register)
✓		qc.mvgeui rd, rs1, imm, rs3	Conditional move if great or equal than unsigned (Immediate)
✓		qc.mvlt rd, rs1, rs2, rs3	Conditional move if less than (Register)
✓		qc.mvlti rd, rs1, imm, rs3	Conditional move if less than (Immediate)

✓		qc.mvltu rd, rs1, rs2, rs3	Conditional move if less than unsigned (Register)
✓		qc.mvltui rd, rs1, imm, rs3	Conditional move if less than unsigned (Immediate)
✓		qc.mvne rd, rs1, rs2, rs3	Conditional move if not equal (Register)
✓		qc.mvnei rd, rs1, imm, rs3	Conditional move if not equal (Immediate)
✓		qc.norm rd, rs1	Signed Normalization
✓		qc.normeu rd, rs1	Unsigned Even Normalization
✓		qc.normu rd, rs1	Unsigned Normalization
✓		qc.outw rs2, imm(rs1)	Output word to non-memory-mapped device
✓		`qc.pcoredump`	Print core dump pseudo-instruction (hint) working only in simulation environment
✓		qc.pexit rs1	Exit call with register argument pseudo-instruction (hint) working only in simulation environment
✓		qc.ppreg rs1	Print register pseudo-instruction (hint) working only in simulation environment
✓		`qc.ppregs`	Print all registers pseudo-instruction (hint) working only in simulation environment
✓		qc.pputc rs1	Print character passed in register argument pseudo-instruction (hint) working only in simulation environment
✓		qc.pputc imm	Print character passed in the immediate argument pseudo-instruction (hint) working only in simulation environment
✓		qc.pputs rs1	Print string pseudo-instruction (hint) working only in simulation environment
✓		qc.psycall rs1	System call with register argument pseudo-instruction (hint) working only in simulation environment
✓		qc.psycalli imm	System call with immediate argument pseudo-instruction (hint) working only in simulation environment
✓		qc.selecteqi rd, imm, rs2, rs3	Select load immediate or register if equal (Immediate)
✓		qc.selectieq rd, rs1, rs2, simm2	Select load immediate or register if equal (Register)
✓		qc.selecteqi rd, imm, rs2, simm2	Select load immediate or register if equal (Immediate)
✓		qc.selectieq rd, rs1, simm1, simm2	Select load immediate if equal (Register)
✓		qc.selectiine rd, rs1, simm1, simm2	Select load immediate if not equal (Register)
✓		qc.selectine rd, rs1, rs2, simm2	Select load immediate or register if not equal (Register)
✓		qc.selectinei rd, imm, rs2, simm2	Select load immediate or register if not equal (Immediate)

✓		qc.selectnei rd, imm, rs2, rs3	Select load immediate or register if not equal (Immediate)
✓		qc.setinti imm	Set interrupt (Immediate)
✓		qc.setwm rs3, rs2, imm(rs1)	Set word multiple (Register)
✓		qc.setwmi rs3, length, imm(rs1)	Set word multiple (Immediate)
✓		qc.shladd rd, rs1, rs2, shamt	Shift left and add (immediate)
✓		qc.shlsat rd, rs1, rs2	Saturating signed left shift
✓		qc.shlusat rd, rs1, rs2	Saturating unsigned left shift
✓		qc.srb rs3, rs1, rs2, shamt	Store indexed byte
✓		qc.srh rs3, rs1, rs2, shamt	Store indexed halfword
✓		qc.srw rs3, rs1, rs2, shamt	Store indexed word
✓		qc.subsat rd, rs1, rs2	Saturating signed subtraction
✓		qc.subusat rd, rs1, rs2	Saturating unsigned subtraction
✓		qc.swm rs3, rs2, imm(rs1)	Store word multiple
✓		qc.swmi rs3, length, imm(rs1)	Store word multiple (immediate)
✓		qc.sync imm	Non-memory-mapped device read-after-write synchronization to completion
✓		qc.syncr imm	Non-memory-mapped device read-after-write synchronization
✓		qc.syncwf imm	Non-memory-mapped device write-after-read synchronization
✓		qc.syncwl imm	Non-memory-mapped device write-after-read synchronization to completion
✓		qc.wrap rd, rs1, rs2	Wraparound (Register)
✓		qc.wrapi rd, rs1, imm	Wraparound (Unsigned Immediate)

3.1. Instructions by sub-extension

Mnemonic	Xqcia	Xqciac	Xqciib	Xqciibm	Xqciicli	Xqciicm	Xqciics	Xqciicsr	Xqciicint	Xqciicio	Xqciiclb	Xqciicli	Xqciiclia	Xqciicilo	Xqciiclsim	Xqciicisim	Xqciicisls	Xqciicisync
qc.addsat	✓																	
qc.addusat	✓																	
qc.beqi			✓															
qc.bgei			✓															

Mnemonic	Xqcia	Xqciac	Xqciabi	Xqciabm	Xqciicli	Xqciicm	Xqciics	Xqciicsr	Xqciint	Xqciio	Xqciilb	Xqciili	Xqciilia	Xqciilo	Xqciism	Xqciisim	Xqciisls	Xqciisync
qc.bgeui			✓															
qc.blti			✓															
qc.bltui			✓															
qc.bnei			✓															
qc.brev32				✓														
qc.c.bexti				✓														
qc.c.bseti				✓														
qc.c.clrint									✓									
qc.c.delay																		✓
qc.c.di									✓									
qc.c.dir									✓									
qc.c.ei									✓									
qc.c.eir									✓									
qc.c.extu				✓														
qc.c.mienester.nest									✓									
qc.c.mienester									✓									
qc.c.mileaveret									✓									
qc.c.mnret									✓									
qc.c.mret									✓									
qc.c.muliadd		✓																
qc.c.mveqz						✓												
qc.c.ptrace																✓		
qc.c.setint									✓									

Mnemonic	Xqcia	Xqciac	Xqciabi	Xqciabm	Xqciabcli	Xqciabcm	Xqciabcs	Xqciabcsr	Xqciabiint	Xqciabio	Xqciablb	Xqciabli	Xqciablia	Xqciabilo	Xqciabls	Xqciablsim	Xqciablsis	Xqciabsync
qc.c.sync																		✓
qc.c.syncr																		✓
qc.c.syncwf																		✓
qc.c.syncwl																		✓
qc.clo				✓														
qc.clrinti									✓									
qc.compress2				✓														
qc.compress3				✓														
qc.csrrwr								✓										
qc.csrrwri								✓										
qc.cto				✓														
qc.e.addai													✓					
qc.e.addi													✓					
qc.e.andai													✓					
qc.e.andi													✓					
qc.e.beqi			✓															
qc.e.bgei			✓															
qc.e.bgeui			✓															
qc.e.blti			✓															
qc.e.bltoi			✓															
qc.e.bnei			✓															
qc.e.j											✓							
qc.e.jal											✓							

Mnemonic	Xqcia	Xqciac	Xqci bi	Xqci b m	Xqci cli	Xqci cm	Xqci cs	Xqci csr	Xqci iint	Xqci iio	Xqci ilb	Xqci ili	Xqci ilia	Xqci ilo	Xqci lsm	Xqci sim	Xqci isls	Xqci sync
qc.e.lb														✓				
qc.e.lbu														✓				
qc.e.lh														✓				
qc.e.lhu														✓				
qc.e.li												✓						
qc.e.lw														✓				
qc.e.orai													✓					
qc.e.ori													✓					
qc.e.sb														✓				
qc.e.sh														✓				
qc.e.sw														✓				
qc.e.xorai													✓					
qc.e.xori													✓					
qc.expand2				✓														
qc.expand3				✓														
qc.ext				✓														
qc.extd				✓														
qc.extdpr				✓														
qc.extdprh				✓														
qc.extdr				✓														
qc.extdu				✓														
qc.extdupr				✓														
qc.extduprh				✓														
qc.extdur				✓														
qc.extu				✓														
qc.insb				✓														
qc.insbh				✓														
qc.insbhr				✓														
qc.insbi				✓														

Mnemonic	Xqcia	Xqciac	Xqciabi	Xqciibm	Xqciicli	Xqciicm	Xqciics	Xqciicsr	Xqciint	Xqciio	Xqciilb	Xqciili	Xqciilia	Xqciilo	Xqciism	Xqciisim	Xqciisls	Xqciisync
qc.insbpr				✓														
qc.insbprh				✓														
qc.insbr				✓														
qc.insbri				✓														
qc.inw										✓								
qc.li												✓						
qc.lieq					✓													
qc.lieqi					✓													
qc.lige					✓													
qc.ligei					✓													
qc.ligeu					✓													
qc.ligeui					✓													
qc.lilt					✓													
qc.lilti					✓													
qc.liltu					✓													
qc.liltui					✓													
qc.line					✓													
qc.linei					✓													
qc.lrb																	✓	
qc.lrbu																	✓	
qc.lrh																	✓	
qc.lrhu																	✓	
qc.lrw																	✓	
qc.lwm															✓			
qc.lwmi															✓			
qc.mulia		✓																
qc.mveq						✓												
qc.mveqi						✓												
qc.mvge						✓												
qc.mvgei						✓												

Mnemonic	Xqcia	Xqciac	Xqci bi	Xqci bm	Xqci cli	Xqci cm	Xqci cs	Xqci csr	Xqci iint	Xqci iio	Xqci ilb	Xqci ili	Xqci ilia	Xqci ilo	Xqci lsm	Xqci sim	Xqci isls	Xqci sync
qc.mvgeu						✓												
qc.mvgeui						✓												
qc.mvlt						✓												
qc.mvlti						✓												
qc.mvltu						✓												
qc.mvltui						✓												
qc.mvne						✓												
qc.mvnei						✓												
qc.norm	✓																	
qc.normeu	✓																	
qc.normu	✓																	
qc.outw										✓								
qc.pcor edump																✓		
qc.pexit																✓		
qc.ppre g																✓		
qc.ppre gs																✓		
qc.pput c																✓		
qc.pput ci																✓		
qc.pput s																✓		
qc.psyc all																✓		
qc.psyc alli																✓		
qc.selec teqj							✓											
qc.selec tieq							✓											
qc.selec tieqj							✓											

Mnemonic	Xqcia	Xqciac	Xqciabi	Xqciabm	Xqciicli	Xqciicm	Xqciics	Xqciicsr	Xqciint	Xqciio	Xqciilb	Xqciili	Xqciilia	Xqciilo	Xqciism	Xqciisim	Xqciisls	Xqciisync
qc.selectiieq							✓											
qc.selectiine							✓											
qc.selectine							✓											
qc.selectinei							✓											
qc.selectnei							✓											
qc.setinti									✓									
qc.setwrm														✓				
qc.setwmi														✓				
qc.shladd		✓																
qc.shlsat	✓																	
qc.shlusat	✓																	
qc.srb																	✓	
qc.srh																	✓	
qc.srw																	✓	
qc.subsat	✓																	
qc.subusat	✓																	
qc.swm														✓				
qc.swmi														✓				
qc.sync																		✓
qc.syncr																		✓
qc.syncwf																		✓
qc.syncwl																		✓
qc.wrap	✓																	
qc.wrapi	✓																	

Chapter 4. CSR summary

The following 62 CSRs are affected by this extension.

RV32	RV64	CSR	Name
✓		qc.mcause	Machine Mode Custom Cause Register
✓		qc.mclcie0	IRQ Enable 0
✓		qc.mclcie1	IRQ Enable 1
✓		qc.mclcie2	IRQ Enable 2
✓		qc.mclcie3	IRQ Enable 3
✓		qc.mclcie4	IRQ Enable 4
✓		qc.mclcie5	IRQ Enable 5
✓		qc.mclcie6	IRQ Enable 6
✓		qc.mclcie7	IRQ Enable 7
✓		qc.mclcivl00	IRQ Level 0
✓		qc.mclcivl01	IRQ Level 1
✓		qc.mclcivl02	IRQ Level 2
✓		qc.mclcivl03	IRQ Level 3
✓		qc.mclcivl04	IRQ Level 4
✓		qc.mclcivl05	IRQ Level 5
✓		qc.mclcivl06	IRQ Level 6
✓		qc.mclcivl07	IRQ Level 7
✓		qc.mclcivl08	IRQ Level 8
✓		qc.mclcivl09	IRQ Level 9
✓		qc.mclcivl10	IRQ Level 10
✓		qc.mclcivl11	IRQ Level 11
✓		qc.mclcivl12	IRQ Level 12
✓		qc.mclcivl13	IRQ Level 13
✓		qc.mclcivl14	IRQ Level 14
✓		qc.mclcivl15	IRQ Level 15
✓		qc.mclcivl16	IRQ Level 16
✓		qc.mclcivl17	IRQ Level 17
✓		qc.mclcivl18	IRQ Level 18
✓		qc.mclcivl19	IRQ Level 19
✓		qc.mclcivl20	IRQ Level 20
✓		qc.mclcivl21	IRQ Level 21
✓		qc.mclcivl22	IRQ Level 22

✓		qc.mclivil23	IRQ Level 23
✓		qc.mclivil24	IRQ Level 24
✓		qc.mclivil25	IRQ Level 25
✓		qc.mclivil26	IRQ Level 26
✓		qc.mclivil27	IRQ Level 27
✓		qc.mclivil28	IRQ Level 28
✓		qc.mclivil29	IRQ Level 29
✓		qc.mclivil30	IRQ Level 30
✓		qc.mclivil31	IRQ Level 31
✓		qc.mclicip0	IRQ Pending 0
✓		qc.mclicip1	IRQ Pending 1
✓		qc.mclicip2	IRQ Pending 2
✓		qc.mclicip3	IRQ Pending 3
✓		qc.mclicip4	IRQ Pending 4
✓		qc.mclicip5	IRQ Pending 5
✓		qc.mclicip6	IRQ Pending 6
✓		qc.mclicip7	IRQ Pending 7
✓		qc.mmcr	Machine Mode Control Register
✓		qc.mntvec	Machine Non-Maskable Interrupt Vector Control
✓		qc.mstkbtopaddr	Machine Stack Bottom Limit
✓		qc.mstktopaddr	Machine Stack Top Limit
✓		qc.mthreadptr	Machine Thread Pointer Register
✓		qc.mwpendaddr0	Watchpoint end address for region 0
✓		qc.mwpendaddr1	Watchpoint end address for region 1
✓		qc.mwpendaddr2	Watchpoint end address for region 2
✓		qc.mwpendaddr3	Watchpoint end address for region 3
✓		qc.mwpstartaddr0	Watchpoint start address for region 0
✓		qc.mwpstartaddr1	Watchpoint start address for region 1
✓		qc.mwpstartaddr2	Watchpoint start address for region 2
✓		qc.mwpstartaddr3	Watchpoint start address for region 3

Chapter 5. Csrs (in alphabetical order)

5.1. qc.mcause

Machine Mode Custom Cause Register

Machine Mode Custom Cause Register / Interrupt context register.

5.1.1. Attributes

Defining Extension	- anyOf: - Xqci, version ≥ 0.7 - Xqciint, version ≥ 0.4
CSR Address	0x7c9
Length	32-bit
Privilege Mode	M

5.1.2. Format

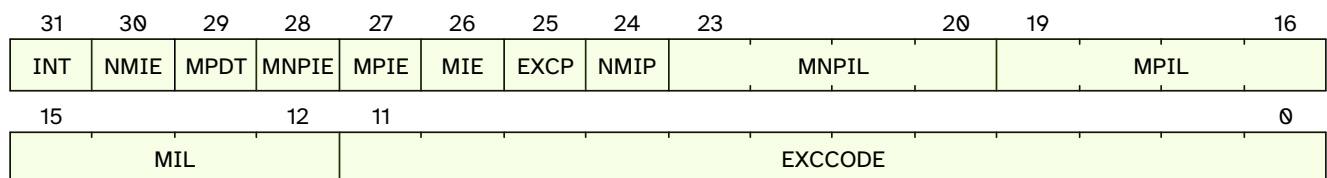


Figure 1. qc.mcause format

5.1.3. Field Summary

Name	Location	Type	Reset Value
qc.mcause.INT	31	RW-H	0
qc.mcause.NMIE	30	RW-H	0
qc.mcause.MPDT	29	RW-H	0
qc.mcause.MNPIE	28	RW-H	0
qc.mcause.MPIE	27	RW-H	0
qc.mcause.MIE	26	RW-H	0
qc.mcause.EXCP	25	RW-H	0
qc.mcause.NMIP	24	RW-H	0
qc.mcause.MNPIL	23:20	RW-H	0
qc.mcause.MPIL	19:16	RW-H	0
qc.mcause.MIL	15:12	RW-H	15
qc.mcause.EXCCODE	11:0	RW-H	0

5.1.4. Fields

INT

Location

31

Description*Alias of mcause.INT bit, if 1'b1 - interrupt processing, if 1'b0 - exception processing.***Type***RW-H***Reset value**

0

NMIE

Location

30

Description*Alias of mnstatus.NMIE, if 1'b0, currently processing NMI or double trap.***Type***RW-H***Reset value**

0

MPDT

Location

29

Description*M-mode Previous Disable Trap, value of mstatush.MDT at the moment of interrupt/exception.***Type***RW-H***Reset value**

0

MNPIE**Location**

28

Description

M-mode NMI Previous Interrupt Enable, value of mstatus.MIE at the moment of NMI/double trap.

Type*RW-H***Reset value**

0

MPIE**Location**

27

Description

Alias of mstatus.MPIE, M-mode Previous Interrupt Enable.

Type*RW-H***Reset value**

0

MIE**Location**

26

Description

Alias of mstatus.MIE, M-mode Interrupt Enable.

Type*RW-H***Reset value**

0

EXCP

Location

25

Description*Exception Pending Bit, cleaned when exception acknowledged.***Type***RW-H***Reset value**

0

NMIP

Location

24

Description*NMI Pending Bit, cleaned when NMI acknowledged.***Type***RW-H***Reset value**

0

MNPIL

Location

23:20

Description*M-mode NMI Previous Interrupt Level, value of qc.mcause.MIL at the moment of NMI/double trap.***Type***RW-H***Reset value**

0

MPIL**Location***19:16***Description***M-mode Previous Interrupt Level, value of qc.mcause.MIL at the moment of interrupt/exception.***Type***RW-H***Reset value***0***MIL****Location***15:12***Description***M-mode Interrupt Level.***Type***RW-H***Reset value***15***EXCCODE****Location***11:0***Description***Alias of mcause.EXCCODE[11:0], Interrupt/Exception code ID.***Type***RW-H***Reset value***0*

5.2. qc.mclcie0

IRQ Enable 0

Enable bits for IRQs 0-31

5.2.1. Attributes

Defining Extension	- anyOf: - Xqci, version >= Xqci@0.1.0 - Xqciint, version >= Xqciint@0.1.0
CSR Address	0x7f8
Length	32-bit
Privilege Mode	M

5.2.2. Format

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
IRQ31	IRQ30	IRQ29	IRQ28	IRQ27	IRQ26	IRQ25	IRQ24	IRQ23	IRQ22	IRQ21	IRQ20	IRQ19	IRQ18	IRQ17	IRQ16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IRQ15	IRQ14	IRQ13	IRQ12	IRQ11	IRQ10	IRQ9	IRQ8	IRQ7	IRQ6	IRQ5	IRQ4	IRQ3	IRQ2	IRQ1	IRQ0

Figure 2. qc.mclcie0 format

5.2.3. Field Summary

Name	Location	Type	Reset Value
qc.mclcie0.IRQ0	0	RW	0
qc.mclcie0.IRQ1	1	RW	0
qc.mclcie0.IRQ2	2	RW	0
qc.mclcie0.IRQ3	3	RW	0
qc.mclcie0.IRQ4	4	RW	0
qc.mclcie0.IRQ5	5	RW	0
qc.mclcie0.IRQ6	6	RW	0
qc.mclcie0.IRQ7	7	RW	0
qc.mclcie0.IRQ8	8	RW	0
qc.mclcie0.IRQ9	9	RW	0
qc.mclcie0.IRQ10	10	RW	0
qc.mclcie0.IRQ11	11	RW	0
qc.mclcie0.IRQ12	12	RW	0
qc.mclcie0.IRQ13	13	RW	0
qc.mclcie0.IRQ14	14	RW	0

Name	Location	Type	Reset Value
qc.mclcie0.IRQ15	15	RW	0
qc.mclcie0.IRQ16	16	RW	0
qc.mclcie0.IRQ17	17	RW	0
qc.mclcie0.IRQ18	18	RW	0
qc.mclcie0.IRQ19	19	RW	0
qc.mclcie0.IRQ20	20	RW	0
qc.mclcie0.IRQ21	21	RW	0
qc.mclcie0.IRQ22	22	RW	0
qc.mclcie0.IRQ23	23	RW	0
qc.mclcie0.IRQ24	24	RW	0
qc.mclcie0.IRQ25	25	RW	0
qc.mclcie0.IRQ26	26	RW	0
qc.mclcie0.IRQ27	27	RW	0
qc.mclcie0.IRQ28	28	RW	0
qc.mclcie0.IRQ29	29	RW	0
qc.mclcie0.IRQ30	30	RW	0
qc.mclcie0.IRQ31	31	RW	0

5.2.4. Fields

IRQ0

Location

0

Description

IRQ0 enabled

Type

RW

Reset value

0

IRQ1

Location

1

Description
<i>IRQ1 enabled</i>
Type
<i>RW</i>
Reset value
0

IRQ2

Location
2
Description
<i>IRQ2 enabled</i>
Type
<i>RW</i>
Reset value
0

IRQ3

Location
3
Description
<i>IRQ3 enabled</i>
Type
<i>RW</i>
Reset value
0

IRQ4

Location
4
Description
<i>IRQ4 enabled</i>

Type*RW***Reset value**

0

IRQ5**Location**

5

Description*IRQ5 enabled***Type***RW***Reset value**

0

IRQ6**Location**

6

Description*IRQ6 enabled***Type***RW***Reset value**

0

IRQ7**Location**

7

Description*IRQ7 enabled***Type***RW*

Reset value
0

IRQ8

Location
8
Description
<i>IRQ8 enabled</i>
Type
<i>RW</i>
Reset value
0

IRQ9

Location
9
Description
<i>IRQ9 enabled</i>
Type
<i>RW</i>
Reset value
0

IRQ10

Location
10
Description
<i>IRQ10 enabled</i>
Type
<i>RW</i>
Reset value
0

IRQ11

Location
11
Description
IRQ11 enabled
Type
RW
Reset value
0

IRQ12

Location
12
Description
IRQ12 enabled
Type
RW
Reset value
0

IRQ13

Location
13
Description
IRQ13 enabled
Type
RW
Reset value
0

IRQ14

Location
14
Description
IRQ14 enabled
Type
RW
Reset value
0

IRQ15

Location
15
Description
IRQ15 enabled
Type
RW
Reset value
0

IRQ16

Location
16
Description
IRQ16 enabled
Type
RW
Reset value
0

IRQ17

Location
17
Description
IRQ17 enabled
Type
RW
Reset value
0

IRQ18

Location
18
Description
IRQ18 enabled
Type
RW
Reset value
0

IRQ19

Location
19
Description
IRQ19 enabled
Type
RW
Reset value
0

IRQ20**Location***20***Description***IRQ20 enabled***Type***RW***Reset value***0***IRQ21****Location***21***Description***IRQ21 enabled***Type***RW***Reset value***0***IRQ22****Location***22***Description***IRQ22 enabled***Type***RW***Reset value***0*

IRQ23

Location
23
Description
<i>IRQ23 enabled</i>
Type
<i>RW</i>
Reset value
0

IRQ24

Location
24
Description
<i>IRQ24 enabled</i>
Type
<i>RW</i>
Reset value
0

IRQ25

Location
25
Description
<i>IRQ25 enabled</i>
Type
<i>RW</i>
Reset value
0

IRQ26**Location**

26

Description*IRQ26 enabled***Type***RW***Reset value**

0

IRQ27**Location**

27

Description*IRQ27 enabled***Type***RW***Reset value**

0

IRQ28**Location**

28

Description*IRQ28 enabled***Type***RW***Reset value**

0

IRQ29

Location
29
Description
IRQ29 enabled
Type
RW
Reset value
0

IRQ30

Location
30
Description
IRQ30 enabled
Type
RW
Reset value
0

IRQ31

Location
31
Description
IRQ31 enabled
Type
RW
Reset value
0

5.3. qc.mclcie1

IRQ Enable 1

Enable bits for IRQs 32-63

5.3.1. Attributes

Defining Extension	- anyOf: - Xqci, version >= Xqci@0.1.0 - Xqciint, version >= Xqciint@0.1.0
CSR Address	0x7f9
Length	32-bit
Privilege Mode	M

5.3.2. Format

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
IRQ63	IRQ62	IRQ61	IRQ60	IRQ59	IRQ58	IRQ57	IRQ56	IRQ55	IRQ54	IRQ53	IRQ52	IRQ51	IRQ50	IRQ49	IRQ48
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IRQ47	IRQ46	IRQ45	IRQ44	IRQ43	IRQ42	IRQ41	IRQ40	IRQ39	IRQ38	IRQ37	IRQ36	IRQ35	IRQ34	IRQ33	IRQ32

Figure 3. qc.mclcie1 format

5.3.3. Field Summary

Name	Location	Type	Reset Value
qc.mclcie1.IRQ32	0	RW	0
qc.mclcie1.IRQ33	1	RW	0
qc.mclcie1.IRQ34	2	RW	0
qc.mclcie1.IRQ35	3	RW	0
qc.mclcie1.IRQ36	4	RW	0
qc.mclcie1.IRQ37	5	RW	0
qc.mclcie1.IRQ38	6	RW	0
qc.mclcie1.IRQ39	7	RW	0
qc.mclcie1.IRQ40	8	RW	0
qc.mclcie1.IRQ41	9	RW	0
qc.mclcie1.IRQ42	10	RW	0
qc.mclcie1.IRQ43	11	RW	0
qc.mclcie1.IRQ44	12	RW	0
qc.mclcie1.IRQ45	13	RW	0
qc.mclcie1.IRQ46	14	RW	0

Name	Location	Type	Reset Value
qc.mclcie1.IRQ47	15	RW	0
qc.mclcie1.IRQ48	16	RW	0
qc.mclcie1.IRQ49	17	RW	0
qc.mclcie1.IRQ50	18	RW	0
qc.mclcie1.IRQ51	19	RW	0
qc.mclcie1.IRQ52	20	RW	0
qc.mclcie1.IRQ53	21	RW	0
qc.mclcie1.IRQ54	22	RW	0
qc.mclcie1.IRQ55	23	RW	0
qc.mclcie1.IRQ56	24	RW	0
qc.mclcie1.IRQ57	25	RW	0
qc.mclcie1.IRQ58	26	RW	0
qc.mclcie1.IRQ59	27	RW	0
qc.mclcie1.IRQ60	28	RW	0
qc.mclcie1.IRQ61	29	RW	0
qc.mclcie1.IRQ62	30	RW	0
qc.mclcie1.IRQ63	31	RW	0

5.3.4. Fields

IRQ32

Location

0

Description

IRQ32 enabled

Type

RW

Reset value

0

IRQ33

Location

1

Description*IRQ33 enabled***Type***RW***Reset value***0***IRQ34****Location***2***Description***IRQ34 enabled***Type***RW***Reset value***0***IRQ35****Location***3***Description***IRQ35 enabled***Type***RW***Reset value***0***IRQ36****Location***4***Description***IRQ36 enabled*

Type*RW***Reset value***0***IRQ37****Location***5***Description***IRQ37 enabled***Type***RW***Reset value***0***IRQ38****Location***6***Description***IRQ38 enabled***Type***RW***Reset value***0***IRQ39****Location***7***Description***IRQ39 enabled***Type***RW*

Reset value
0

IRQ40

Location
8
Description
<i>IRQ40 enabled</i>
Type
<i>RW</i>
Reset value
0

IRQ41

Location
9
Description
<i>IRQ41 enabled</i>
Type
<i>RW</i>
Reset value
0

IRQ42

Location
10
Description
<i>IRQ42 enabled</i>
Type
<i>RW</i>
Reset value
0

IRQ43**Location***11***Description***IRQ43 enabled***Type***RW***Reset value***0***IRQ44****Location***12***Description***IRQ44 enabled***Type***RW***Reset value***0***IRQ45****Location***13***Description***IRQ45 enabled***Type***RW***Reset value***0*

IRQ46**Location***14***Description***IRQ46 enabled***Type***RW***Reset value***0***IRQ47****Location***15***Description***IRQ47 enabled***Type***RW***Reset value***0***IRQ48****Location***16***Description***IRQ48 enabled***Type***RW***Reset value***0*

IRQ49

Location*17***Description***IRQ49 enabled***Type***RW***Reset value***0*

IRQ50

Location*18***Description***IRQ50 enabled***Type***RW***Reset value***0*

IRQ51

Location*19***Description***IRQ51 enabled***Type***RW***Reset value***0*

IRQ52**Location**

20

Description*IRQ52 enabled***Type***RW***Reset value**

0

IRQ53**Location**

21

Description*IRQ53 enabled***Type***RW***Reset value**

0

IRQ54**Location**

22

Description*IRQ54 enabled***Type***RW***Reset value**

0

IRQ55

Location*23***Description***IRQ55 enabled***Type***RW***Reset value***0*

IRQ56

Location*24***Description***IRQ56 enabled***Type***RW***Reset value***0*

IRQ57

Location*25***Description***IRQ57 enabled***Type***RW***Reset value***0*

IRQ58**Location**

26

Description*IRQ58 enabled***Type***RW***Reset value**

0

IRQ59**Location**

27

Description*IRQ59 enabled***Type***RW***Reset value**

0

IRQ60**Location**

28

Description*IRQ60 enabled***Type***RW***Reset value**

0

IRQ61

Location*29***Description***IRQ61 enabled***Type***RW***Reset value***0*

IRQ62

Location*30***Description***IRQ62 enabled***Type***RW***Reset value***0*

IRQ63

Location*31***Description***IRQ63 enabled***Type***RW***Reset value***0*

5.4. qc.mclicie2

IRQ Enable 2

Enable bits for IRQs 64-95

5.4.1. Attributes

Defining Extension	- anyOf: - Xqci, version >= Xqci@0.1.0 - Xqciint, version >= Xqciint@0.1.0
CSR Address	0x7fa
Length	32-bit
Privilege Mode	M

5.4.2. Format

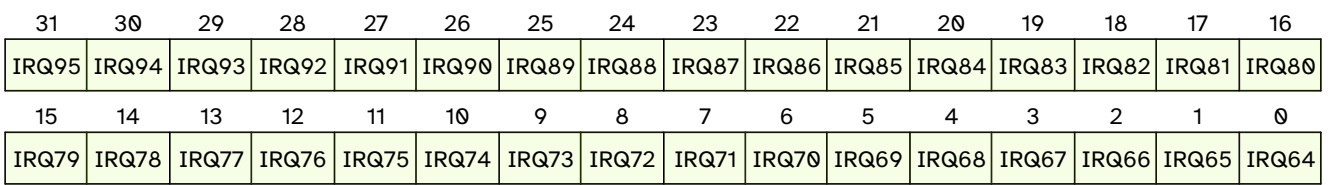


Figure 4. qc.mclicie2 format

5.4.3. Field Summary

Name	Location	Type	Reset Value
qc.mclicie2.IRQ64	0	RW	0
qc.mclicie2.IRQ65	1	RW	0
qc.mclicie2.IRQ66	2	RW	0
qc.mclicie2.IRQ67	3	RW	0
qc.mclicie2.IRQ68	4	RW	0
qc.mclicie2.IRQ69	5	RW	0
qc.mclicie2.IRQ70	6	RW	0
qc.mclicie2.IRQ71	7	RW	0
qc.mclicie2.IRQ72	8	RW	0
qc.mclicie2.IRQ73	9	RW	0
qc.mclicie2.IRQ74	10	RW	0
qc.mclicie2.IRQ75	11	RW	0
qc.mclicie2.IRQ76	12	RW	0
qc.mclicie2.IRQ77	13	RW	0
qc.mclicie2.IRQ78	14	RW	0

Name	Location	Type	Reset Value
qc.mclcie2.IRQ79	15	RW	0
qc.mclcie2.IRQ80	16	RW	0
qc.mclcie2.IRQ81	17	RW	0
qc.mclcie2.IRQ82	18	RW	0
qc.mclcie2.IRQ83	19	RW	0
qc.mclcie2.IRQ84	20	RW	0
qc.mclcie2.IRQ85	21	RW	0
qc.mclcie2.IRQ86	22	RW	0
qc.mclcie2.IRQ87	23	RW	0
qc.mclcie2.IRQ88	24	RW	0
qc.mclcie2.IRQ89	25	RW	0
qc.mclcie2.IRQ90	26	RW	0
qc.mclcie2.IRQ91	27	RW	0
qc.mclcie2.IRQ92	28	RW	0
qc.mclcie2.IRQ93	29	RW	0
qc.mclcie2.IRQ94	30	RW	0
qc.mclcie2.IRQ95	31	RW	0

5.4.4. Fields

IRQ64

Location

0

Description

IRQ64 enabled

Type

RW

Reset value

0

IRQ65

Location

1

Description*IRQ65 enabled***Type***RW***Reset value***0***IRQ66****Location***2***Description***IRQ66 enabled***Type***RW***Reset value***0***IRQ67****Location***3***Description***IRQ67 enabled***Type***RW***Reset value***0***IRQ68****Location***4***Description***IRQ68 enabled*

Type*RW***Reset value**

0

IRQ69**Location**

5

Description*IRQ69 enabled***Type***RW***Reset value**

0

IRQ70**Location**

6

Description*IRQ70 enabled***Type***RW***Reset value**

0

IRQ71**Location**

7

Description*IRQ71 enabled***Type***RW*

Reset value
0

IRQ72

Location
8
Description
<i>IRQ72 enabled</i>
Type
<i>RW</i>
Reset value
0

IRQ73

Location
9
Description
<i>IRQ73 enabled</i>
Type
<i>RW</i>
Reset value
0

IRQ74

Location
10
Description
<i>IRQ74 enabled</i>
Type
<i>RW</i>
Reset value
0

IRQ75**Location***11***Description***IRQ75 enabled***Type***RW***Reset value***0***IRQ76****Location***12***Description***IRQ76 enabled***Type***RW***Reset value***0***IRQ77****Location***13***Description***IRQ77 enabled***Type***RW***Reset value***0*

IRQ78**Location***14***Description***IRQ78 enabled***Type***RW***Reset value***0***IRQ79****Location***15***Description***IRQ79 enabled***Type***RW***Reset value***0***IRQ80****Location***16***Description***IRQ80 enabled***Type***RW***Reset value***0*

IRQ81**Location***17***Description***IRQ81 enabled***Type***RW***Reset value***0***IRQ82****Location***18***Description***IRQ82 enabled***Type***RW***Reset value***0***IRQ83****Location***19***Description***IRQ83 enabled***Type***RW***Reset value***0*

IRQ84

Location

20

Description*IRQ84 enabled***Type***RW***Reset value**

0

IRQ85

Location

21

Description*IRQ85 enabled***Type***RW***Reset value**

0

IRQ86

Location

22

Description*IRQ86 enabled***Type***RW***Reset value**

0

IRQ87

Location

23

Description*IRQ87 enabled***Type***RW***Reset value**

0

IRQ88

Location

24

Description*IRQ88 enabled***Type***RW***Reset value**

0

IRQ89

Location

25

Description*IRQ89 enabled***Type***RW***Reset value**

0

IRQ90**Location**

26

Description*IRQ90 enabled***Type***RW***Reset value**

0

IRQ91**Location**

27

Description*IRQ91 enabled***Type***RW***Reset value**

0

IRQ92**Location**

28

Description*IRQ92 enabled***Type***RW***Reset value**

0

IRQ93**Location***29***Description***IRQ93 enabled***Type***RW***Reset value***0***IRQ94****Location***30***Description***IRQ94 enabled***Type***RW***Reset value***0***IRQ95****Location***31***Description***IRQ95 enabled***Type***RW***Reset value***0*

5.5. qc.mclicie3

IRQ Enable 3

Enable bits for IRQs 96-127

5.5.1. Attributes

Defining Extension	- anyOf: - Xqci, version >= Xqci@0.1.0 - Xqciint, version >= Xqciint@0.1.0
CSR Address	0x7fb
Length	32-bit
Privilege Mode	M

5.5.2. Format

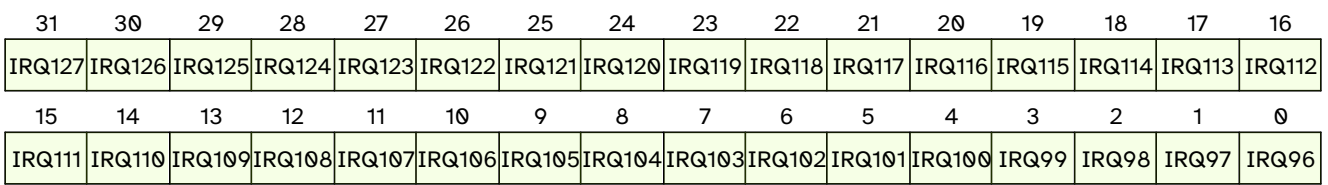


Figure 5. qc.mclicie3 format

5.5.3. Field Summary

Name	Location	Type	Reset Value
qc.mclicie3.IRQ96	0	RW	0
qc.mclicie3.IRQ97	1	RW	0
qc.mclicie3.IRQ98	2	RW	0
qc.mclicie3.IRQ99	3	RW	0
qc.mclicie3.IRQ100	4	RW	0
qc.mclicie3.IRQ101	5	RW	0
qc.mclicie3.IRQ102	6	RW	0
qc.mclicie3.IRQ103	7	RW	0
qc.mclicie3.IRQ104	8	RW	0
qc.mclicie3.IRQ105	9	RW	0
qc.mclicie3.IRQ106	10	RW	0
qc.mclicie3.IRQ107	11	RW	0
qc.mclicie3.IRQ108	12	RW	0
qc.mclicie3.IRQ109	13	RW	0
qc.mclicie3.IRQ110	14	RW	0

Name	Location	Type	Reset Value
qc.mclcie3.IRQ111	15	RW	0
qc.mclcie3.IRQ112	16	RW	0
qc.mclcie3.IRQ113	17	RW	0
qc.mclcie3.IRQ114	18	RW	0
qc.mclcie3.IRQ115	19	RW	0
qc.mclcie3.IRQ116	20	RW	0
qc.mclcie3.IRQ117	21	RW	0
qc.mclcie3.IRQ118	22	RW	0
qc.mclcie3.IRQ119	23	RW	0
qc.mclcie3.IRQ120	24	RW	0
qc.mclcie3.IRQ121	25	RW	0
qc.mclcie3.IRQ122	26	RW	0
qc.mclcie3.IRQ123	27	RW	0
qc.mclcie3.IRQ124	28	RW	0
qc.mclcie3.IRQ125	29	RW	0
qc.mclcie3.IRQ126	30	RW	0
qc.mclcie3.IRQ127	31	RW	0

5.5.4. Fields

IRQ96

Location

0

Description

IRQ96 enabled

Type

RW

Reset value

0

IRQ97

Location

1

Description*IRQ97 enabled***Type***RW***Reset value***0***IRQ98****Location***2***Description***IRQ98 enabled***Type***RW***Reset value***0***IRQ99****Location***3***Description***IRQ99 enabled***Type***RW***Reset value***0***IRQ100****Location***4***Description***IRQ100 enabled*

Type*RW***Reset value***0***IRQ101****Location***5***Description***IRQ101 enabled***Type***RW***Reset value***0***IRQ102****Location***6***Description***IRQ102 enabled***Type***RW***Reset value***0***IRQ103****Location***7***Description***IRQ103 enabled***Type***RW*

Reset value

0

IRQ104**Location**

8

Description*IRQ104 enabled***Type***RW***Reset value**

0

IRQ105**Location**

9

Description*IRQ105 enabled***Type***RW***Reset value**

0

IRQ106**Location**

10

Description*IRQ106 enabled***Type***RW***Reset value**

0

IRQ107**Location***11***Description***IRQ107 enabled***Type***RW***Reset value***0***IRQ108****Location***12***Description***IRQ108 enabled***Type***RW***Reset value***0***IRQ109****Location***13***Description***IRQ109 enabled***Type***RW***Reset value***0*

IRQ110**Location***14***Description***IRQ110 enabled***Type***RW***Reset value***0***IRQ111****Location***15***Description***IRQ111 enabled***Type***RW***Reset value***0***IRQ112****Location***16***Description***IRQ112 enabled***Type***RW***Reset value***0*

IRQ113**Location***17***Description***IRQ113 enabled***Type***RW***Reset value***0***IRQ114****Location***18***Description***IRQ114 enabled***Type***RW***Reset value***0***IRQ115****Location***19***Description***IRQ115 enabled***Type***RW***Reset value***0*

IRQ116**Location***20***Description***IRQ116 enabled***Type***RW***Reset value***0***IRQ117****Location***21***Description***IRQ117 enabled***Type***RW***Reset value***0***IRQ118****Location***22***Description***IRQ118 enabled***Type***RW***Reset value***0*

IRQ119**Location***23***Description***IRQ119 enabled***Type***RW***Reset value***0***IRQ120****Location***24***Description***IRQ120 enabled***Type***RW***Reset value***0***IRQ121****Location***25***Description***IRQ121 enabled***Type***RW***Reset value***0*

IRQ122**Location**

26

Description*IRQ122 enabled***Type***RW***Reset value**

0

IRQ123**Location**

27

Description*IRQ123 enabled***Type***RW***Reset value**

0

IRQ124**Location**

28

Description*IRQ124 enabled***Type***RW***Reset value**

0

IRQ125

Location*29***Description***IRQ125 enabled***Type***RW***Reset value***0*

IRQ126

Location*30***Description***IRQ126 enabled***Type***RW***Reset value***0*

IRQ127

Location*31***Description***IRQ127 enabled***Type***RW***Reset value***0*

5.6. qc.mclicie4

IRQ Enable 4

Enable bits for IRQs 128-159

5.6.1. Attributes

Defining Extension	- anyOf: - Xqci, version >= Xqci@0.1.0 - Xqciint, version >= Xqciint@0.1.0
CSR Address	0x7fc
Length	32-bit
Privilege Mode	M

5.6.2. Format

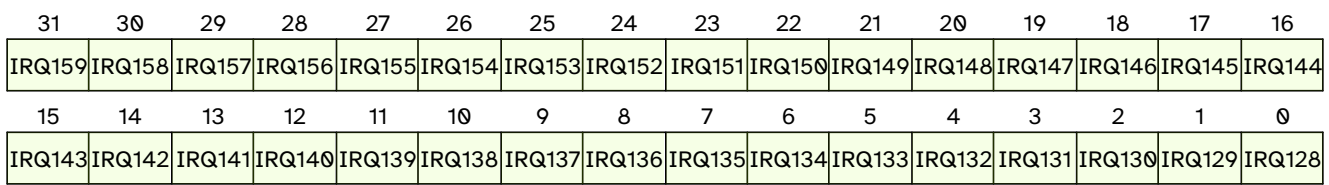


Figure 6. qc.mclicie4 format

5.6.3. Field Summary

Name	Location	Type	Reset Value
qc.mclicie4.IRQ128	0	RW	0
qc.mclicie4.IRQ129	1	RW	0
qc.mclicie4.IRQ130	2	RW	0
qc.mclicie4.IRQ131	3	RW	0
qc.mclicie4.IRQ132	4	RW	0
qc.mclicie4.IRQ133	5	RW	0
qc.mclicie4.IRQ134	6	RW	0
qc.mclicie4.IRQ135	7	RW	0
qc.mclicie4.IRQ136	8	RW	0
qc.mclicie4.IRQ137	9	RW	0
qc.mclicie4.IRQ138	10	RW	0
qc.mclicie4.IRQ139	11	RW	0
qc.mclicie4.IRQ140	12	RW	0
qc.mclicie4.IRQ141	13	RW	0
qc.mclicie4.IRQ142	14	RW	0

Name	Location	Type	Reset Value
qc.mclcie4.IRQ143	15	RW	0
qc.mclcie4.IRQ144	16	RW	0
qc.mclcie4.IRQ145	17	RW	0
qc.mclcie4.IRQ146	18	RW	0
qc.mclcie4.IRQ147	19	RW	0
qc.mclcie4.IRQ148	20	RW	0
qc.mclcie4.IRQ149	21	RW	0
qc.mclcie4.IRQ150	22	RW	0
qc.mclcie4.IRQ151	23	RW	0
qc.mclcie4.IRQ152	24	RW	0
qc.mclcie4.IRQ153	25	RW	0
qc.mclcie4.IRQ154	26	RW	0
qc.mclcie4.IRQ155	27	RW	0
qc.mclcie4.IRQ156	28	RW	0
qc.mclcie4.IRQ157	29	RW	0
qc.mclcie4.IRQ158	30	RW	0
qc.mclcie4.IRQ159	31	RW	0

5.6.4. Fields

IRQ128

Location

0

Description

IRQ128 enabled

Type

RW

Reset value

0

IRQ129

Location

1

Description*IRQ129 enabled***Type***RW***Reset value**

0

IRQ130**Location**

2

Description*IRQ130 enabled***Type***RW***Reset value**

0

IRQ131**Location**

3

Description*IRQ131 enabled***Type***RW***Reset value**

0

IRQ132**Location**

4

Description*IRQ132 enabled*

Type*RW***Reset value**

0

IRQ133**Location**

5

Description*IRQ133 enabled***Type***RW***Reset value**

0

IRQ134**Location**

6

Description*IRQ134 enabled***Type***RW***Reset value**

0

IRQ135**Location**

7

Description*IRQ135 enabled***Type***RW*

Reset value

0

IRQ136**Location**

8

Description*IRQ136 enabled***Type***RW***Reset value**

0

IRQ137**Location**

9

Description*IRQ137 enabled***Type***RW***Reset value**

0

IRQ138**Location**

10

Description*IRQ138 enabled***Type***RW***Reset value**

0

IRQ139**Location***11***Description***IRQ139 enabled***Type***RW***Reset value***0***IRQ140****Location***12***Description***IRQ140 enabled***Type***RW***Reset value***0***IRQ141****Location***13***Description***IRQ141 enabled***Type***RW***Reset value***0*

IRQ142**Location***14***Description***IRQ142 enabled***Type***RW***Reset value***0***IRQ143****Location***15***Description***IRQ143 enabled***Type***RW***Reset value***0***IRQ144****Location***16***Description***IRQ144 enabled***Type***RW***Reset value***0*

IRQ145**Location***17***Description***IRQ145 enabled***Type***RW***Reset value***0***IRQ146****Location***18***Description***IRQ146 enabled***Type***RW***Reset value***0***IRQ147****Location***19***Description***IRQ147 enabled***Type***RW***Reset value***0*

IRQ148**Location***20***Description***IRQ148 enabled***Type***RW***Reset value***0***IRQ149****Location***21***Description***IRQ149 enabled***Type***RW***Reset value***0***IRQ150****Location***22***Description***IRQ150 enabled***Type***RW***Reset value***0*

IRQ151**Location***23***Description***IRQ151 enabled***Type***RW***Reset value***0***IRQ152****Location***24***Description***IRQ152 enabled***Type***RW***Reset value***0***IRQ153****Location***25***Description***IRQ153 enabled***Type***RW***Reset value***0*

IRQ154**Location**

26

Description*IRQ154 enabled***Type***RW***Reset value**

0

IRQ155**Location**

27

Description*IRQ155 enabled***Type***RW***Reset value**

0

IRQ156**Location**

28

Description*IRQ156 enabled***Type***RW***Reset value**

0

IRQ157**Location***29***Description***IRQ157 enabled***Type***RW***Reset value***0***IRQ158****Location***30***Description***IRQ158 enabled***Type***RW***Reset value***0***IRQ159****Location***31***Description***IRQ159 enabled***Type***RW***Reset value***0*

5.7. qc.mclcie5

IRQ Enable 5

Enable bits for IRQs 160-191

5.7.1. Attributes

Defining Extension	- anyOf: - Xqci, version >= Xqci@0.1.0 - Xqciint, version >= Xqciint@0.1.0
CSR Address	0x7fd
Length	32-bit
Privilege Mode	M

5.7.2. Format

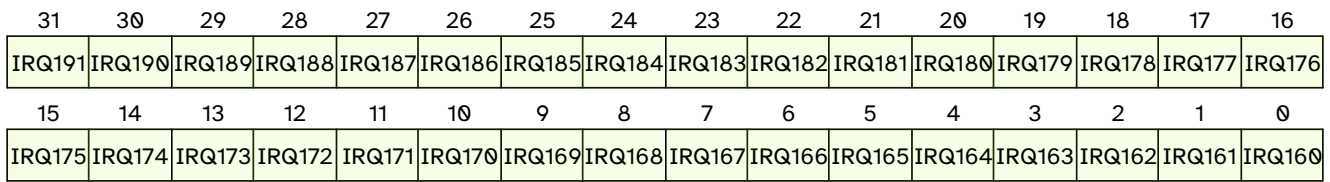


Figure 7. qc.mclcie5 format

5.7.3. Field Summary

Name	Location	Type	Reset Value
qc.mclcie5.IRQ160	0	RW	0
qc.mclcie5.IRQ161	1	RW	0
qc.mclcie5.IRQ162	2	RW	0
qc.mclcie5.IRQ163	3	RW	0
qc.mclcie5.IRQ164	4	RW	0
qc.mclcie5.IRQ165	5	RW	0
qc.mclcie5.IRQ166	6	RW	0
qc.mclcie5.IRQ167	7	RW	0
qc.mclcie5.IRQ168	8	RW	0
qc.mclcie5.IRQ169	9	RW	0
qc.mclcie5.IRQ170	10	RW	0
qc.mclcie5.IRQ171	11	RW	0
qc.mclcie5.IRQ172	12	RW	0
qc.mclcie5.IRQ173	13	RW	0
qc.mclcie5.IRQ174	14	RW	0

Name	Location	Type	Reset Value
qc.mclcie5.IRQ175	15	RW	0
qc.mclcie5.IRQ176	16	RW	0
qc.mclcie5.IRQ177	17	RW	0
qc.mclcie5.IRQ178	18	RW	0
qc.mclcie5.IRQ179	19	RW	0
qc.mclcie5.IRQ180	20	RW	0
qc.mclcie5.IRQ181	21	RW	0
qc.mclcie5.IRQ182	22	RW	0
qc.mclcie5.IRQ183	23	RW	0
qc.mclcie5.IRQ184	24	RW	0
qc.mclcie5.IRQ185	25	RW	0
qc.mclcie5.IRQ186	26	RW	0
qc.mclcie5.IRQ187	27	RW	0
qc.mclcie5.IRQ188	28	RW	0
qc.mclcie5.IRQ189	29	RW	0
qc.mclcie5.IRQ190	30	RW	0
qc.mclcie5.IRQ191	31	RW	0

5.7.4. Fields

IRQ160

Location

0

Description

IRQ160 enabled

Type

RW

Reset value

0

IRQ161

Location

1

Description*IRQ161 enabled***Type***RW***Reset value***0***IRQ162****Location***2***Description***IRQ162 enabled***Type***RW***Reset value***0***IRQ163****Location***3***Description***IRQ163 enabled***Type***RW***Reset value***0***IRQ164****Location***4***Description***IRQ164 enabled*

Type*RW***Reset value**

0

IRQ165**Location**

5

Description*IRQ165 enabled***Type***RW***Reset value**

0

IRQ166**Location**

6

Description*IRQ166 enabled***Type***RW***Reset value**

0

IRQ167**Location**

7

Description*IRQ167 enabled***Type***RW*

Reset value

0

IRQ168**Location**

8

Description*IRQ168 enabled***Type***RW***Reset value**

0

IRQ169**Location**

9

Description*IRQ169 enabled***Type***RW***Reset value**

0

IRQ170**Location**

10

Description*IRQ170 enabled***Type***RW***Reset value**

0

IRQ171**Location***11***Description***IRQ171 enabled***Type***RW***Reset value***0***IRQ172****Location***12***Description***IRQ172 enabled***Type***RW***Reset value***0***IRQ173****Location***13***Description***IRQ173 enabled***Type***RW***Reset value***0*

IRQ174**Location***14***Description***IRQ174 enabled***Type***RW***Reset value***0***IRQ175****Location***15***Description***IRQ175 enabled***Type***RW***Reset value***0***IRQ176****Location***16***Description***IRQ176 enabled***Type***RW***Reset value***0*

IRQ177**Location***17***Description***IRQ177 enabled***Type***RW***Reset value***0***IRQ178****Location***18***Description***IRQ178 enabled***Type***RW***Reset value***0***IRQ179****Location***19***Description***IRQ179 enabled***Type***RW***Reset value***0*

IRQ180**Location***20***Description***IRQ180 enabled***Type***RW***Reset value***0***IRQ181****Location***21***Description***IRQ181 enabled***Type***RW***Reset value***0***IRQ182****Location***22***Description***IRQ182 enabled***Type***RW***Reset value***0*

IRQ183**Location***23***Description***IRQ183 enabled***Type***RW***Reset value***0***IRQ184****Location***24***Description***IRQ184 enabled***Type***RW***Reset value***0***IRQ185****Location***25***Description***IRQ185 enabled***Type***RW***Reset value***0*

IRQ186**Location**

26

Description*IRQ186 enabled***Type***RW***Reset value**

0

IRQ187**Location**

27

Description*IRQ187 enabled***Type***RW***Reset value**

0

IRQ188**Location**

28

Description*IRQ188 enabled***Type***RW***Reset value**

0

IRQ189**Location***29***Description***IRQ189 enabled***Type***RW***Reset value***0***IRQ190****Location***30***Description***IRQ190 enabled***Type***RW***Reset value***0***IRQ191****Location***31***Description***IRQ191 enabled***Type***RW***Reset value***0*

5.8. qc.mclicie6

IRQ Enable 6

Enable bits for IRQs 192-223

5.8.1. Attributes

Defining Extension	- anyOf: - Xqci, version >= Xqci@0.1.0 - Xqciint, version >= Xqciint@0.1.0
CSR Address	0x7fe
Length	32-bit
Privilege Mode	M

5.8.2. Format

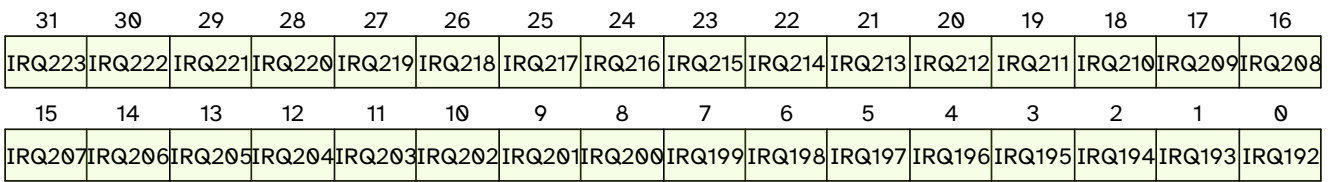


Figure 8. qc.mclicie6 format

5.8.3. Field Summary

Name	Location	Type	Reset Value
qc.mclicie6.IRQ192	0	RW	0
qc.mclicie6.IRQ193	1	RW	0
qc.mclicie6.IRQ194	2	RW	0
qc.mclicie6.IRQ195	3	RW	0
qc.mclicie6.IRQ196	4	RW	0
qc.mclicie6.IRQ197	5	RW	0
qc.mclicie6.IRQ198	6	RW	0
qc.mclicie6.IRQ199	7	RW	0
qc.mclicie6.IRQ200	8	RW	0
qc.mclicie6.IRQ201	9	RW	0
qc.mclicie6.IRQ202	10	RW	0
qc.mclicie6.IRQ203	11	RW	0
qc.mclicie6.IRQ204	12	RW	0
qc.mclicie6.IRQ205	13	RW	0
qc.mclicie6.IRQ206	14	RW	0

Name	Location	Type	Reset Value
qc.mclcie6.IRQ207	15	RW	0
qc.mclcie6.IRQ208	16	RW	0
qc.mclcie6.IRQ209	17	RW	0
qc.mclcie6.IRQ210	18	RW	0
qc.mclcie6.IRQ211	19	RW	0
qc.mclcie6.IRQ212	20	RW	0
qc.mclcie6.IRQ213	21	RW	0
qc.mclcie6.IRQ214	22	RW	0
qc.mclcie6.IRQ215	23	RW	0
qc.mclcie6.IRQ216	24	RW	0
qc.mclcie6.IRQ217	25	RW	0
qc.mclcie6.IRQ218	26	RW	0
qc.mclcie6.IRQ219	27	RW	0
qc.mclcie6.IRQ220	28	RW	0
qc.mclcie6.IRQ221	29	RW	0
qc.mclcie6.IRQ222	30	RW	0
qc.mclcie6.IRQ223	31	RW	0

5.8.4. Fields

IRQ192

Location

0

Description

IRQ192 enabled

Type

RW

Reset value

0

IRQ193

Location

1

Description

IRQ193 enabled

Type

RW

Reset value

0

IRQ194

Location

2

Description

IRQ194 enabled

Type

RW

Reset value

0

IRQ195

Location

3

Description

IRQ195 enabled

Type

RW

Reset value

0

IRQ196

Location

4

Description

IRQ196 enabled

Type*RW***Reset value**

0

IRQ197**Location**

5

Description*IRQ197 enabled***Type***RW***Reset value**

0

IRQ198**Location**

6

Description*IRQ198 enabled***Type***RW***Reset value**

0

IRQ199**Location**

7

Description*IRQ199 enabled***Type***RW*

Reset value
0

IRQ200

Location
8
Description
<i>IRQ200 enabled</i>
Type
<i>RW</i>
Reset value
0

IRQ201

Location
9
Description
<i>IRQ201 enabled</i>
Type
<i>RW</i>
Reset value
0

IRQ202

Location
10
Description
<i>IRQ202 enabled</i>
Type
<i>RW</i>
Reset value
0

IRQ203**Location***11***Description***IRQ203 enabled***Type***RW***Reset value***0***IRQ204****Location***12***Description***IRQ204 enabled***Type***RW***Reset value***0***IRQ205****Location***13***Description***IRQ205 enabled***Type***RW***Reset value***0*

IRQ206**Location***14***Description***IRQ206 enabled***Type***RW***Reset value***0***IRQ207****Location***15***Description***IRQ207 enabled***Type***RW***Reset value***0***IRQ208****Location***16***Description***IRQ208 enabled***Type***RW***Reset value***0*

IRQ209**Location***17***Description***IRQ209 enabled***Type***RW***Reset value***0***IRQ210****Location***18***Description***IRQ210 enabled***Type***RW***Reset value***0***IRQ211****Location***19***Description***IRQ211 enabled***Type***RW***Reset value***0*

IRQ212**Location***20***Description***IRQ212 enabled***Type***RW***Reset value***0***IRQ213****Location***21***Description***IRQ213 enabled***Type***RW***Reset value***0***IRQ214****Location***22***Description***IRQ214 enabled***Type***RW***Reset value***0*

IRQ215

Location*23***Description***IRQ215 enabled***Type***RW***Reset value***0*

IRQ216

Location*24***Description***IRQ216 enabled***Type***RW***Reset value***0*

IRQ217

Location*25***Description***IRQ217 enabled***Type***RW***Reset value***0*

IRQ218**Location**

26

Description*IRQ218 enabled***Type***RW***Reset value**

0

IRQ219**Location**

27

Description*IRQ219 enabled***Type***RW***Reset value**

0

IRQ220**Location**

28

Description*IRQ220 enabled***Type***RW***Reset value**

0

IRQ221

Location*29***Description***IRQ221 enabled***Type***RW***Reset value***0*

IRQ222

Location*30***Description***IRQ222 enabled***Type***RW***Reset value***0*

IRQ223

Location*31***Description***IRQ223 enabled***Type***RW***Reset value***0*

5.9. qc.mclcie7

IRQ Enable 7

Enable bits for IRQs 224-255

5.9.1. Attributes

Defining Extension	- anyOf: - Xqci, version >= Xqci@0.1.0 - Xqciint, version >= Xqciint@0.1.0
CSR Address	0x7ff
Length	32-bit
Privilege Mode	M

5.9.2. Format

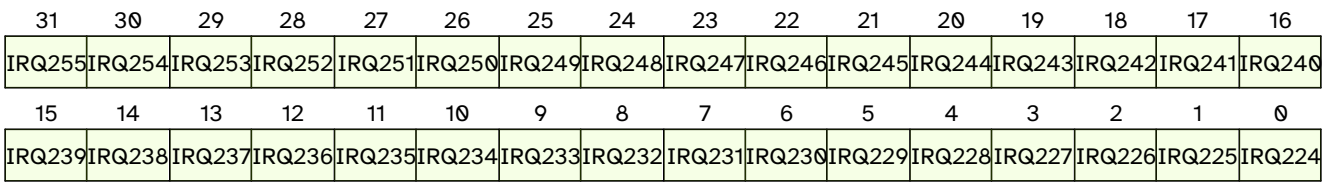


Figure 9. qc.mclcie7 format

5.9.3. Field Summary

Name	Location	Type	Reset Value
qc.mclcie7.IRQ224	0	RW	0
qc.mclcie7.IRQ225	1	RW	0
qc.mclcie7.IRQ226	2	RW	0
qc.mclcie7.IRQ227	3	RW	0
qc.mclcie7.IRQ228	4	RW	0
qc.mclcie7.IRQ229	5	RW	0
qc.mclcie7.IRQ230	6	RW	0
qc.mclcie7.IRQ231	7	RW	0
qc.mclcie7.IRQ232	8	RW	0
qc.mclcie7.IRQ233	9	RW	0
qc.mclcie7.IRQ234	10	RW	0
qc.mclcie7.IRQ235	11	RW	0
qc.mclcie7.IRQ236	12	RW	0
qc.mclcie7.IRQ237	13	RW	0
qc.mclcie7.IRQ238	14	RW	0

Name	Location	Type	Reset Value
qc.mclcie7.IRQ239	15	RW	0
qc.mclcie7.IRQ240	16	RW	0
qc.mclcie7.IRQ241	17	RW	0
qc.mclcie7.IRQ242	18	RW	0
qc.mclcie7.IRQ243	19	RW	0
qc.mclcie7.IRQ244	20	RW	0
qc.mclcie7.IRQ245	21	RW	0
qc.mclcie7.IRQ246	22	RW	0
qc.mclcie7.IRQ247	23	RW	0
qc.mclcie7.IRQ248	24	RW	0
qc.mclcie7.IRQ249	25	RW	0
qc.mclcie7.IRQ250	26	RW	0
qc.mclcie7.IRQ251	27	RW	0
qc.mclcie7.IRQ252	28	RW	0
qc.mclcie7.IRQ253	29	RW	0
qc.mclcie7.IRQ254	30	RW	0
qc.mclcie7.IRQ255	31	RW	0

5.9.4. Fields

IRQ224

Location

0

Description

IRQ224 enabled

Type

RW

Reset value

0

IRQ225

Location

1

Description*IRQ225 enabled***Type***RW***Reset value***0***IRQ226****Location***2***Description***IRQ226 enabled***Type***RW***Reset value***0***IRQ227****Location***3***Description***IRQ227 enabled***Type***RW***Reset value***0***IRQ228****Location***4***Description***IRQ228 enabled*

Type*RW***Reset value***0***IRQ229****Location***5***Description***IRQ229 enabled***Type***RW***Reset value***0***IRQ230****Location***6***Description***IRQ230 enabled***Type***RW***Reset value***0***IRQ231****Location***7***Description***IRQ231 enabled***Type***RW*

Reset value

0

IRQ232**Location**

8

Description*IRQ232 enabled***Type***RW***Reset value**

0

IRQ233**Location**

9

Description*IRQ233 enabled***Type***RW***Reset value**

0

IRQ234**Location**

10

Description*IRQ234 enabled***Type***RW***Reset value**

0

IRQ235

Location*11***Description***IRQ235 enabled***Type***RW***Reset value***0*

IRQ236

Location*12***Description***IRQ236 enabled***Type***RW***Reset value***0*

IRQ237

Location*13***Description***IRQ237 enabled***Type***RW***Reset value***0*

IRQ238**Location***14***Description***IRQ238 enabled***Type***RW***Reset value***0***IRQ239****Location***15***Description***IRQ239 enabled***Type***RW***Reset value***0***IRQ240****Location***16***Description***IRQ240 enabled***Type***RW***Reset value***0*

IRQ241**Location***17***Description***IRQ241 enabled***Type***RW***Reset value***0***IRQ242****Location***18***Description***IRQ242 enabled***Type***RW***Reset value***0***IRQ243****Location***19***Description***IRQ243 enabled***Type***RW***Reset value***0*

IRQ244**Location***20***Description***IRQ244 enabled***Type***RW***Reset value***0***IRQ245****Location***21***Description***IRQ245 enabled***Type***RW***Reset value***0***IRQ246****Location***22***Description***IRQ246 enabled***Type***RW***Reset value***0*

IRQ247

Location*23***Description***IRQ247 enabled***Type***RW***Reset value***0*

IRQ248

Location*24***Description***IRQ248 enabled***Type***RW***Reset value***0*

IRQ249

Location*25***Description***IRQ249 enabled***Type***RW***Reset value***0*

IRQ250**Location**

26

Description*IRQ250 enabled***Type***RW***Reset value**

0

IRQ251**Location**

27

Description*IRQ251 enabled***Type***RW***Reset value**

0

IRQ252**Location**

28

Description*IRQ252 enabled***Type***RW***Reset value**

0

IRQ253

Location*29***Description***IRQ253 enabled***Type***RW***Reset value***0*

IRQ254

Location*30***Description***IRQ254 enabled***Type***RW***Reset value***0*

IRQ255

Location*31***Description***IRQ255 enabled***Type***RW***Reset value***0*

5.10. qc.mclcivl00

IRQ Level 0

Level bits for IRQs 0-7

5.10.1. Attributes

Defining Extension	- anyOf: - Xqci, version >=0.7 - Xqciint, version >=0.4
CSR Address	0xbc0
Length	32-bit
Privilege Mode	M

5.10.2. Format

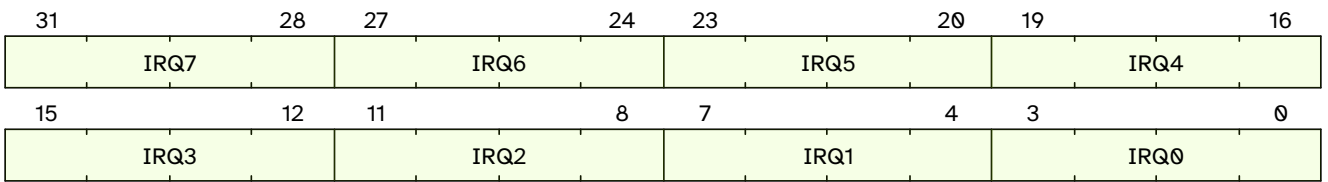


Figure 10. qc.mclcivl00 format

5.10.3. Field Summary

Name	Location	Type	Reset Value
qc.mclcivl00.IRQ0	3:0	RW	0
qc.mclcivl00.IRQ1	7:4	RW	0
qc.mclcivl00.IRQ2	11:8	RW	0
qc.mclcivl00.IRQ3	15:12	RW	0
qc.mclcivl00.IRQ4	19:16	RW	0
qc.mclcivl00.IRQ5	23:20	RW	0
qc.mclcivl00.IRQ6	27:24	RW	0
qc.mclcivl00.IRQ7	31:28	RW	0

5.10.4. Fields

IRQ0

Location
3:0

Description

IRQ0 level

Type

RW

Reset value

0

IRQ1

Location

7:4

Description

IRQ1 level

Type

RW

Reset value

0

IRQ2

Location

11:8

Description

IRQ2 level

Type

RW

Reset value

0

IRQ3

Location

15:12

Description

IRQ3 level

Type
<i>RW</i>
Reset value
<i>0</i>

IRQ4

Location
<i>19:16</i>
Description
<i>IRQ4 level</i>
Type
<i>RW</i>
Reset value
<i>0</i>

IRQ5

Location
<i>23:20</i>
Description
<i>IRQ5 level</i>
Type
<i>RW</i>
Reset value
<i>0</i>

IRQ6

Location
<i>27:24</i>
Description
<i>IRQ6 level</i>
Type
<i>RW</i>

Reset value

0

IRQ7

Location

31:28

Description

IRQ7 level

Type

RW

Reset value

0

5.11. qc.mclcivl01

IRQ Level 1

Level bits for IRQs 8-15

5.11.1. Attributes

Defining Extension	- anyOf: - Xqci, version >=0.7 - Xqciint, version >=0.4
CSR Address	0xbc1
Length	32-bit
Privilege Mode	M

5.11.2. Format

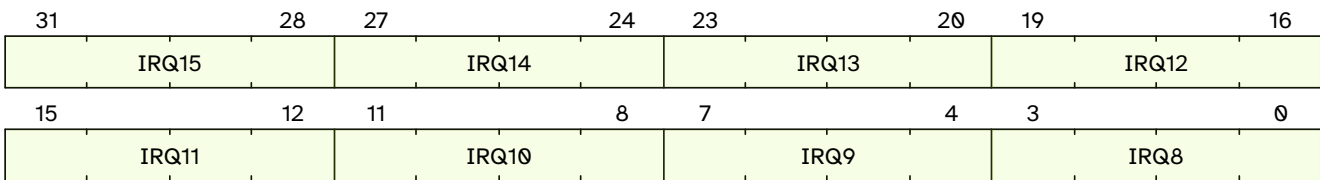


Figure 11. qc.mclcivl01 format

5.11.3. Field Summary

Name	Location	Type	Reset Value
qc.mclcivl01.IRQ8	3:0	RW	0
qc.mclcivl01.IRQ9	7:4	RW	0
qc.mclcivl01.IRQ10	11:8	RW	0
qc.mclcivl01.IRQ11	15:12	RW	0
qc.mclcivl01.IRQ12	19:16	RW	0
qc.mclcivl01.IRQ13	23:20	RW	0
qc.mclcivl01.IRQ14	27:24	RW	0
qc.mclcivl01.IRQ15	31:28	RW	0

5.11.4. Fields

IRQ8

Location
3:0

Description*IRQ8 level***Type***RW***Reset value***0***IRQ9****Location***7:4***Description***IRQ9 level***Type***RW***Reset value***0***IRQ10****Location***11:8***Description***IRQ10 level***Type***RW***Reset value***0***IRQ11****Location***15:12***Description***IRQ11 level*

Type*RW***Reset value**

0

IRQ12**Location***19:16***Description***IRQ12 level***Type***RW***Reset value**

0

IRQ13**Location***23:20***Description***IRQ13 level***Type***RW***Reset value**

0

IRQ14**Location***27:24***Description***IRQ14 level***Type***RW*

Reset value

0

IRQ15

Location

31:28

Description

IRQ15 level

Type

RW

Reset value

0

5.12. qc.mclcivl02

IRQ Level 2

Level bits for IRQs 16-23

5.12.1. Attributes

Defining Extension	- anyOf: - Xqci, version >=0.7 - Xqciint, version >=0.4
CSR Address	0xbc2
Length	32-bit
Privilege Mode	M

5.12.2. Format

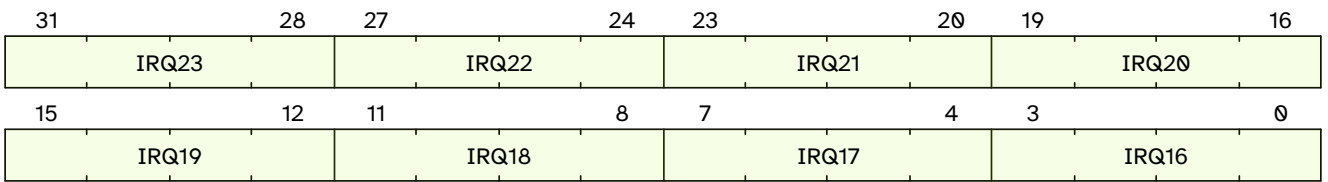


Figure 12. qc.mclcivl02 format

5.12.3. Field Summary

Name	Location	Type	Reset Value
qc.mclcivl02.IRQ16	3:0	RW	0
qc.mclcivl02.IRQ17	7:4	RW	0
qc.mclcivl02.IRQ18	11:8	RW	0
qc.mclcivl02.IRQ19	15:12	RW	0
qc.mclcivl02.IRQ20	19:16	RW	0
qc.mclcivl02.IRQ21	23:20	RW	0
qc.mclcivl02.IRQ22	27:24	RW	0
qc.mclcivl02.IRQ23	31:28	RW	0

5.12.4. Fields

IRQ16

Location
3:0

Description*IRQ16 level***Type***RW***Reset value***0***IRQ17****Location***7:4***Description***IRQ17 level***Type***RW***Reset value***0***IRQ18****Location***11:8***Description***IRQ18 level***Type***RW***Reset value***0***IRQ19****Location***15:12***Description***IRQ19 level*

Type*RW***Reset value***0***IRQ20****Location***19:16***Description***IRQ20 level***Type***RW***Reset value***0***IRQ21****Location***23:20***Description***IRQ21 level***Type***RW***Reset value***0***IRQ22****Location***27:24***Description***IRQ22 level***Type***RW*

Reset value

0

IRQ23**Location**

31:28

Description*IRQ23 level***Type***RW***Reset value**

0

5.13. qc.mclcivl03

IRQ Level 3

Level bits for IRQs 24-31

5.13.1. Attributes

Defining Extension	- anyOf: - Xqci, version >=0.7 - Xqciint, version >=0.4
CSR Address	0xbc3
Length	32-bit
Privilege Mode	M

5.13.2. Format

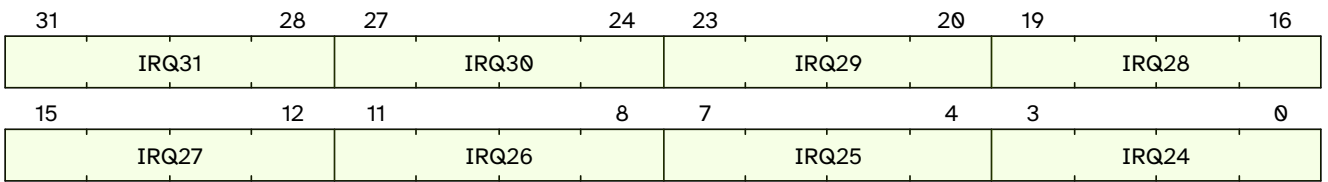


Figure 13. qc.mclcivl03 format

5.13.3. Field Summary

Name	Location	Type	Reset Value
qc.mclcivl03.IRQ24	3:0	RW	0
qc.mclcivl03.IRQ25	7:4	RW	0
qc.mclcivl03.IRQ26	11:8	RW	0
qc.mclcivl03.IRQ27	15:12	RW	0
qc.mclcivl03.IRQ28	19:16	RW	0
qc.mclcivl03.IRQ29	23:20	RW	0
qc.mclcivl03.IRQ30	27:24	RW	0
qc.mclcivl03.IRQ31	31:28	RW	0

5.13.4. Fields

IRQ24

Location
3:0

Description*IRQ24 level***Type***RW***Reset value***0***IRQ25****Location***7:4***Description***IRQ25 level***Type***RW***Reset value***0***IRQ26****Location***11:8***Description***IRQ26 level***Type***RW***Reset value***0***IRQ27****Location***15:12***Description***IRQ27 level*

Type*RW***Reset value**

0

IRQ28**Location***19:16***Description***IRQ28 level***Type***RW***Reset value**

0

IRQ29**Location***23:20***Description***IRQ29 level***Type***RW***Reset value**

0

IRQ30**Location***27:24***Description***IRQ30 level***Type***RW*

Reset value*0***IRQ31****Location***31:28***Description***IRQ31 level***Type***RW***Reset value***0*

5.14. qc.mclcivl04

IRQ Level 4

Level bits for IRQs 32-39

5.14.1. Attributes

Defining Extension	- anyOf: - Xqci, version >=0.7 - Xqciint, version >=0.4
CSR Address	0xbc4
Length	32-bit
Privilege Mode	M

5.14.2. Format

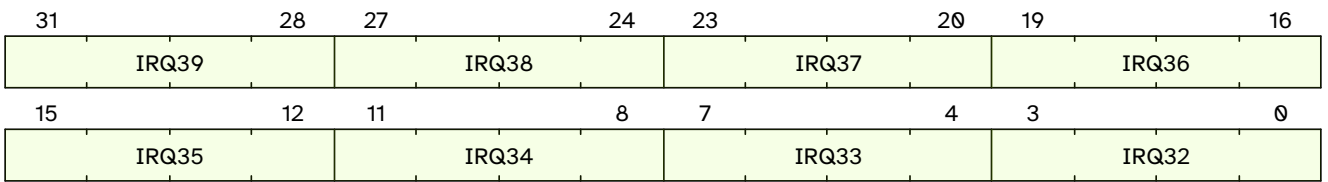


Figure 14. qc.mclcivl04 format

5.14.3. Field Summary

Name	Location	Type	Reset Value
qc.mclcivl04.IRQ32	3:0	RW	0
qc.mclcivl04.IRQ33	7:4	RW	0
qc.mclcivl04.IRQ34	11:8	RW	0
qc.mclcivl04.IRQ35	15:12	RW	0
qc.mclcivl04.IRQ36	19:16	RW	0
qc.mclcivl04.IRQ37	23:20	RW	0
qc.mclcivl04.IRQ38	27:24	RW	0
qc.mclcivl04.IRQ39	31:28	RW	0

5.14.4. Fields

IRQ32

Location
3:0

Description*IRQ32 level***Type***RW***Reset value***0***IRQ33****Location***7:4***Description***IRQ33 level***Type***RW***Reset value***0***IRQ34****Location***11:8***Description***IRQ34 level***Type***RW***Reset value***0***IRQ35****Location***15:12***Description***IRQ35 level*

Type*RW***Reset value***0***IRQ36****Location***19:16***Description***IRQ36 level***Type***RW***Reset value***0***IRQ37****Location***23:20***Description***IRQ37 level***Type***RW***Reset value***0***IRQ38****Location***27:24***Description***IRQ38 level***Type***RW*

Reset value

0

IRQ39

Location

31:28

Description

IRQ39 level

Type

RW

Reset value

0

5.15. qc.mclcivl05

IRQ Level 5

Level bits for IRQs 40-47

5.15.1. Attributes

Defining Extension	- anyOf: - Xqci, version >=0.7 - Xqciint, version >=0.4
CSR Address	0xbc5
Length	32-bit
Privilege Mode	M

5.15.2. Format

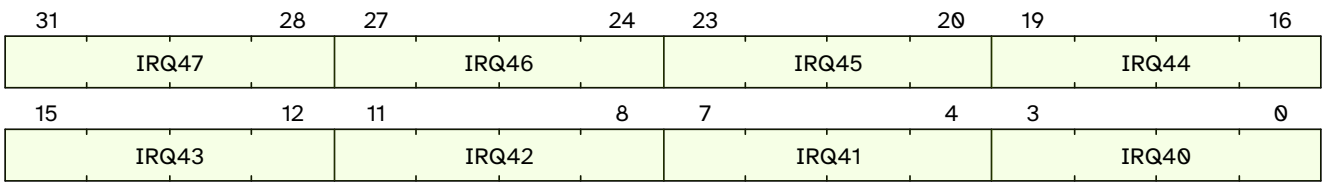


Figure 15. qc.mclcivl05 format

5.15.3. Field Summary

Name	Location	Type	Reset Value
qc.mclcivl05.IRQ40	3:0	RW	0
qc.mclcivl05.IRQ41	7:4	RW	0
qc.mclcivl05.IRQ42	11:8	RW	0
qc.mclcivl05.IRQ43	15:12	RW	0
qc.mclcivl05.IRQ44	19:16	RW	0
qc.mclcivl05.IRQ45	23:20	RW	0
qc.mclcivl05.IRQ46	27:24	RW	0
qc.mclcivl05.IRQ47	31:28	RW	0

5.15.4. Fields

IRQ40

Location

3:0

Description*IRQ40 level***Type***RW***Reset value***0***IRQ41****Location***7:4***Description***IRQ41 level***Type***RW***Reset value***0***IRQ42****Location***11:8***Description***IRQ42 level***Type***RW***Reset value***0***IRQ43****Location***15:12***Description***IRQ43 level*

Type*RW***Reset value***0***IRQ44****Location***19:16***Description***IRQ44 level***Type***RW***Reset value***0***IRQ45****Location***23:20***Description***IRQ45 level***Type***RW***Reset value***0***IRQ46****Location***27:24***Description***IRQ46 level***Type***RW*

Reset value

0

IRQ47**Location**

31:28

Description*IRQ47 level***Type***RW***Reset value**

0

5.16. qc.mclcivl06

IRQ Level 6

Level bits for IRQs 48-55

5.16.1. Attributes

Defining Extension	- anyOf: - Xqci, version >=0.7 - Xqciint, version >=0.4
CSR Address	0xbc6
Length	32-bit
Privilege Mode	M

5.16.2. Format

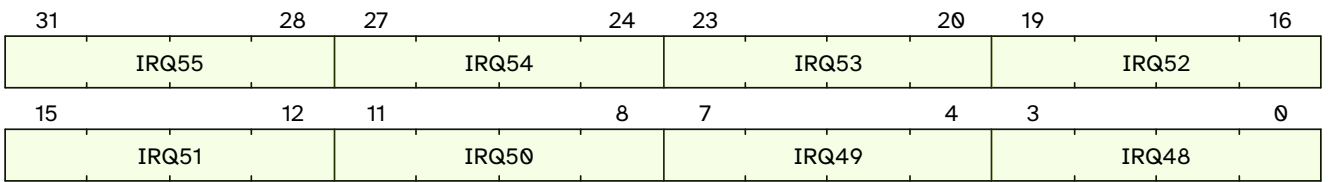


Figure 16. qc.mclcivl06 format

5.16.3. Field Summary

Name	Location	Type	Reset Value
qc.mclcivl06.IRQ48	3:0	RW	0
qc.mclcivl06.IRQ49	7:4	RW	0
qc.mclcivl06.IRQ50	11:8	RW	0
qc.mclcivl06.IRQ51	15:12	RW	0
qc.mclcivl06.IRQ52	19:16	RW	0
qc.mclcivl06.IRQ53	23:20	RW	0
qc.mclcivl06.IRQ54	27:24	RW	0
qc.mclcivl06.IRQ55	31:28	RW	0

5.16.4. Fields

IRQ48

Location
3:0

Description*IRQ48 level***Type***RW***Reset value**

0

IRQ49**Location***7:4***Description***IRQ49 level***Type***RW***Reset value**

0

IRQ50**Location***11:8***Description***IRQ50 level***Type***RW***Reset value**

0

IRQ51**Location***15:12***Description***IRQ51 level*

Type*RW***Reset value**

0

IRQ52**Location***19:16***Description***IRQ52 level***Type***RW***Reset value**

0

IRQ53**Location***23:20***Description***IRQ53 level***Type***RW***Reset value**

0

IRQ54**Location***27:24***Description***IRQ54 level***Type***RW*

Reset value

0

IRQ55**Location**

31:28

Description*IRQ55 level***Type***RW***Reset value**

0

5.17. qc.mclcivl07

IRQ Level 7

Level bits for IRQs 56-63

5.17.1. Attributes

Defining Extension	- anyOf: - Xqci, version >=0.7 - Xqciint, version >=0.4
CSR Address	0xbc7
Length	32-bit
Privilege Mode	M

5.17.2. Format

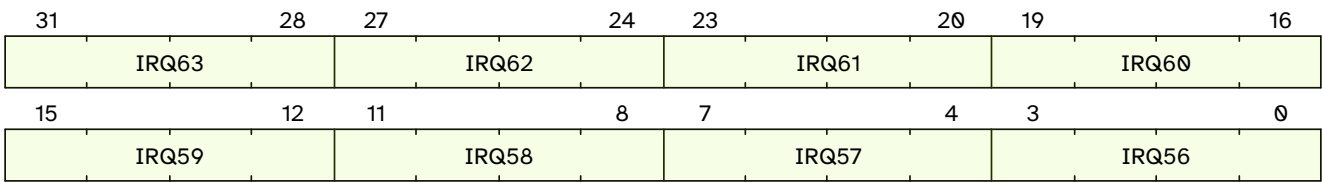


Figure 17. qc.mclcivl07 format

5.17.3. Field Summary

Name	Location	Type	Reset Value
qc.mclcivl07.IRQ56	3:0	RW	0
qc.mclcivl07.IRQ57	7:4	RW	0
qc.mclcivl07.IRQ58	11:8	RW	0
qc.mclcivl07.IRQ59	15:12	RW	0
qc.mclcivl07.IRQ60	19:16	RW	0
qc.mclcivl07.IRQ61	23:20	RW	0
qc.mclcivl07.IRQ62	27:24	RW	0
qc.mclcivl07.IRQ63	31:28	RW	0

5.17.4. Fields

IRQ56

Location

3:0

Description*IRQ56 level***Type***RW***Reset value***0***IRQ57****Location***7:4***Description***IRQ57 level***Type***RW***Reset value***0***IRQ58****Location***11:8***Description***IRQ58 level***Type***RW***Reset value***0***IRQ59****Location***15:12***Description***IRQ59 level*

Type*RW***Reset value***0***IRQ60****Location***19:16***Description***IRQ60 level***Type***RW***Reset value***0***IRQ61****Location***23:20***Description***IRQ61 level***Type***RW***Reset value***0***IRQ62****Location***27:24***Description***IRQ62 level***Type***RW*

Reset value

0

IRQ63**Location**

31:28

Description*IRQ63 level***Type***RW***Reset value**

0

5.18. qc.mclcivl08

IRQ Level 8

Level bits for IRQs 64-71

5.18.1. Attributes

Defining Extension	- anyOf: - Xqci, version >=0.7 - Xqciint, version >=0.4
CSR Address	0xbc8
Length	32-bit
Privilege Mode	M

5.18.2. Format

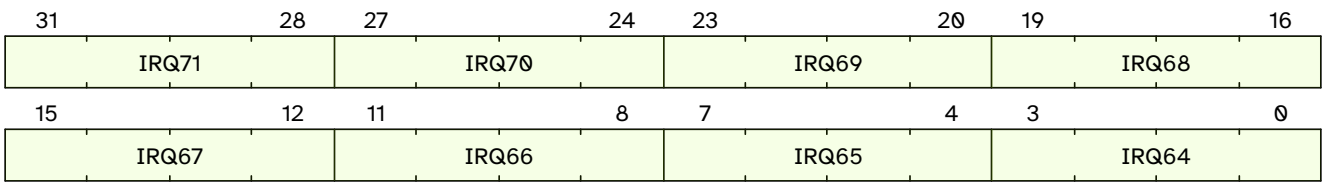


Figure 18. qc.mclcivl08 format

5.18.3. Field Summary

Name	Location	Type	Reset Value
qc.mclcivl08.IRQ64	3:0	RW	0
qc.mclcivl08.IRQ65	7:4	RW	0
qc.mclcivl08.IRQ66	11:8	RW	0
qc.mclcivl08.IRQ67	15:12	RW	0
qc.mclcivl08.IRQ68	19:16	RW	0
qc.mclcivl08.IRQ69	23:20	RW	0
qc.mclcivl08.IRQ70	27:24	RW	0
qc.mclcivl08.IRQ71	31:28	RW	0

5.18.4. Fields

IRQ64

Location

3:0

Description*IRQ64 level***Type***RW***Reset value***0***IRQ65****Location***7:4***Description***IRQ65 level***Type***RW***Reset value***0***IRQ66****Location***11:8***Description***IRQ66 level***Type***RW***Reset value***0***IRQ67****Location***15:12***Description***IRQ67 level*

Type*RW***Reset value***0***IRQ68****Location***19:16***Description***IRQ68 level***Type***RW***Reset value***0***IRQ69****Location***23:20***Description***IRQ69 level***Type***RW***Reset value***0***IRQ70****Location***27:24***Description***IRQ70 level***Type***RW*

Reset value

0

IRQ71

Location

31:28

Description

IRQ71 level

Type

RW

Reset value

0

5.19. qc.mclcivl09

IRQ Level 9

Level bits for IRQs 72-79

5.19.1. Attributes

Defining Extension	- anyOf: - Xqci, version >=0.7 - Xqciint, version >=0.4
CSR Address	0xbc9
Length	32-bit
Privilege Mode	M

5.19.2. Format

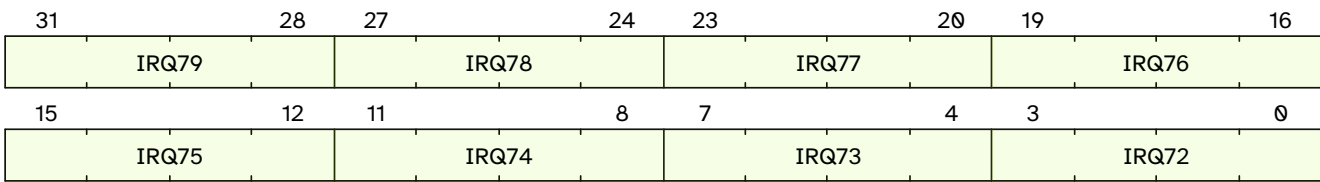


Figure 19. qc.mclcivl09 format

5.19.3. Field Summary

Name	Location	Type	Reset Value
qc.mclcivl09.IRQ72	3:0	RW	0
qc.mclcivl09.IRQ73	7:4	RW	0
qc.mclcivl09.IRQ74	11:8	RW	0
qc.mclcivl09.IRQ75	15:12	RW	0
qc.mclcivl09.IRQ76	19:16	RW	0
qc.mclcivl09.IRQ77	23:20	RW	0
qc.mclcivl09.IRQ78	27:24	RW	0
qc.mclcivl09.IRQ79	31:28	RW	0

5.19.4. Fields

IRQ72

Location

3:0

Description*IRQ72 level***Type***RW***Reset value***0***IRQ73****Location***7:4***Description***IRQ73 level***Type***RW***Reset value***0***IRQ74****Location***11:8***Description***IRQ74 level***Type***RW***Reset value***0***IRQ75****Location***15:12***Description***IRQ75 level*

Type*RW***Reset value**

0

IRQ76**Location***19:16***Description***IRQ76 level***Type***RW***Reset value**

0

IRQ77**Location***23:20***Description***IRQ77 level***Type***RW***Reset value**

0

IRQ78**Location***27:24***Description***IRQ78 level***Type***RW*

Reset value

0

IRQ79**Location**

31:28

Description*IRQ79 level***Type***RW***Reset value**

0

5.20. qc.mclcivl10

IRQ Level 10

Level bits for IRQs 80-87

5.20.1. Attributes

Defining Extension	- anyOf: - Xqci, version >=0.7 - Xqciint, version >=0.4
CSR Address	0xbca
Length	32-bit
Privilege Mode	M

5.20.2. Format

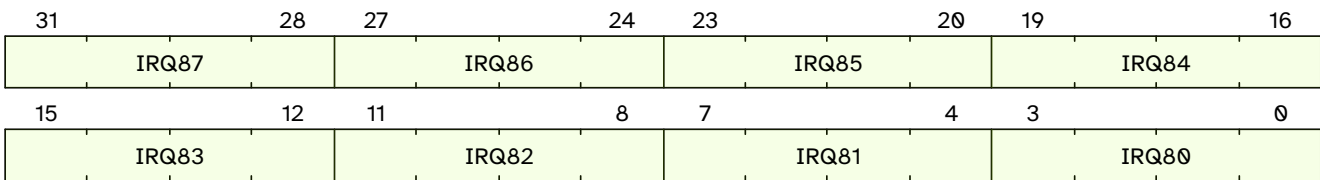


Figure 20. qc.mclcivl10 format

5.20.3. Field Summary

Name	Location	Type	Reset Value
qc.mclcivl10.IRQ80	3:0	RW	0
qc.mclcivl10.IRQ81	7:4	RW	0
qc.mclcivl10.IRQ82	11:8	RW	0
qc.mclcivl10.IRQ83	15:12	RW	0
qc.mclcivl10.IRQ84	19:16	RW	0
qc.mclcivl10.IRQ85	23:20	RW	0
qc.mclcivl10.IRQ86	27:24	RW	0
qc.mclcivl10.IRQ87	31:28	RW	0

5.20.4. Fields

IRQ80

Location

3:0

Description*IRQ80 level***Type***RW***Reset value***0***IRQ81****Location***7:4***Description***IRQ81 level***Type***RW***Reset value***0***IRQ82****Location***11:8***Description***IRQ82 level***Type***RW***Reset value***0***IRQ83****Location***15:12***Description***IRQ83 level*

Type*RW***Reset value***0***IRQ84****Location***19:16***Description***IRQ84 level***Type***RW***Reset value***0***IRQ85****Location***23:20***Description***IRQ85 level***Type***RW***Reset value***0***IRQ86****Location***27:24***Description***IRQ86 level***Type***RW*

Reset value

0

IRQ87**Location**

31:28

Description*IRQ87 level***Type***RW***Reset value**

0

5.21. qc.mclcivl11

IRQ Level 11

Level bits for IRQs 88-95

5.21.1. Attributes

Defining Extension	- anyOf: - Xqci, version >=0.7 - Xqciint, version >=0.4
CSR Address	0xbcb
Length	32-bit
Privilege Mode	M

5.21.2. Format

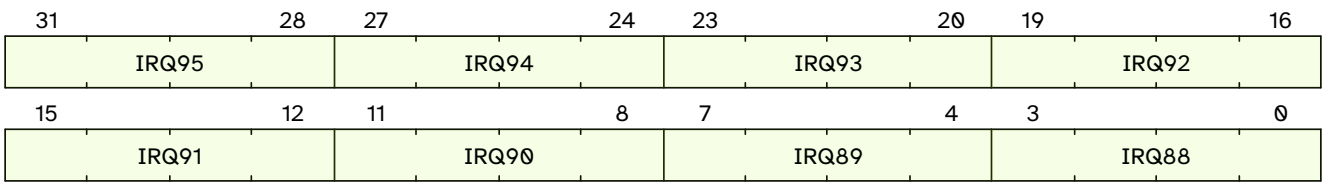


Figure 21. qc.mclcivl11 format

5.21.3. Field Summary

Name	Location	Type	Reset Value
qc.mclcivl11.IRQ88	3:0	RW	0
qc.mclcivl11.IRQ89	7:4	RW	0
qc.mclcivl11.IRQ90	11:8	RW	0
qc.mclcivl11.IRQ91	15:12	RW	0
qc.mclcivl11.IRQ92	19:16	RW	0
qc.mclcivl11.IRQ93	23:20	RW	0
qc.mclcivl11.IRQ94	27:24	RW	0
qc.mclcivl11.IRQ95	31:28	RW	0

5.21.4. Fields

IRQ88

Location
3:0

Description*IRQ88 level***Type***RW***Reset value***0***IRQ89****Location***7:4***Description***IRQ89 level***Type***RW***Reset value***0***IRQ90****Location***11:8***Description***IRQ90 level***Type***RW***Reset value***0***IRQ91****Location***15:12***Description***IRQ91 level*

Type*RW***Reset value***0***IRQ92****Location***19:16***Description***IRQ92 level***Type***RW***Reset value***0***IRQ93****Location***23:20***Description***IRQ93 level***Type***RW***Reset value***0***IRQ94****Location***27:24***Description***IRQ94 level***Type***RW*

Reset value

0

IRQ95

Location

31:28

Description

IRQ95 level

Type

RW

Reset value

0

5.22. qc.mclcivl12

IRQ Level 12

Level bits for IRQs 96-103

5.22.1. Attributes

Defining Extension	- anyOf: - Xqci, version >=0.7 - Xqciint, version >=0.4
CSR Address	0xbcc
Length	32-bit
Privilege Mode	M

5.22.2. Format

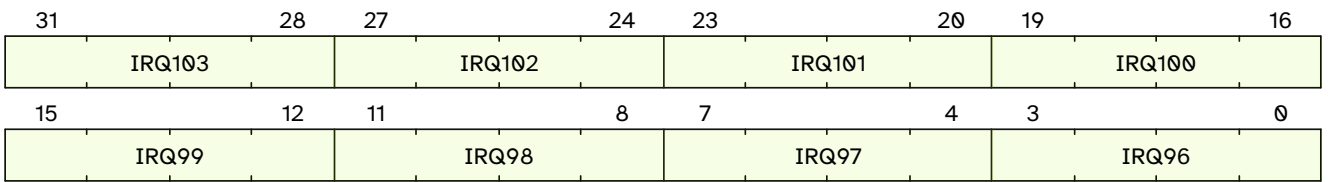


Figure 22. qc.mclcivl12 format

5.22.3. Field Summary

Name	Location	Type	Reset Value
qc.mclcivl12.IRQ96	3:0	RW	0
qc.mclcivl12.IRQ97	7:4	RW	0
qc.mclcivl12.IRQ98	11:8	RW	0
qc.mclcivl12.IRQ99	15:12	RW	0
qc.mclcivl12.IRQ100	19:16	RW	0
qc.mclcivl12.IRQ101	23:20	RW	0
qc.mclcivl12.IRQ102	27:24	RW	0
qc.mclcivl12.IRQ103	31:28	RW	0

5.22.4. Fields

IRQ96

Location
3:0

Description*IRQ96 level***Type***RW***Reset value***0***IRQ97****Location***7:4***Description***IRQ97 level***Type***RW***Reset value***0***IRQ98****Location***11:8***Description***IRQ98 level***Type***RW***Reset value***0***IRQ99****Location***15:12***Description***IRQ99 level*

Type*RW***Reset value**

0

IRQ100**Location***19:16***Description***IRQ100 level***Type***RW***Reset value**

0

IRQ101**Location***23:20***Description***IRQ101 level***Type***RW***Reset value**

0

IRQ102**Location***27:24***Description***IRQ102 level***Type***RW*

Reset value

0

IRQ103**Location**

31:28

Description*IRQ103 level***Type***RW***Reset value**

0

5.23. qc.mclcivl13

IRQ Level 13

Level bits for IRQs 104-111

5.23.1. Attributes

Defining Extension	- anyOf: - Xqci, version >=0.7 - Xqciint, version >=0.4
CSR Address	0xbcd
Length	32-bit
Privilege Mode	M

5.23.2. Format

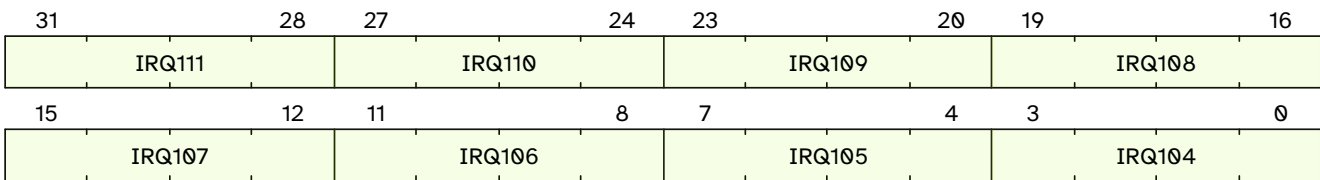


Figure 23. qc.mclcivl13 format

5.23.3. Field Summary

Name	Location	Type	Reset Value
qc.mclcivl13.IRQ104	3:0	RW	0
qc.mclcivl13.IRQ105	7:4	RW	0
qc.mclcivl13.IRQ106	11:8	RW	0
qc.mclcivl13.IRQ107	15:12	RW	0
qc.mclcivl13.IRQ108	19:16	RW	0
qc.mclcivl13.IRQ109	23:20	RW	0
qc.mclcivl13.IRQ110	27:24	RW	0
qc.mclcivl13.IRQ111	31:28	RW	0

5.23.4. Fields

IRQ104

Location
3:0

Description*IRQ104 level***Type***RW***Reset value***0***IRQ105****Location***7:4***Description***IRQ105 level***Type***RW***Reset value***0***IRQ106****Location***11:8***Description***IRQ106 level***Type***RW***Reset value***0***IRQ107****Location***15:12***Description***IRQ107 level*

Type*RW***Reset value***0***IRQ108****Location***19:16***Description***IRQ108 level***Type***RW***Reset value***0***IRQ109****Location***23:20***Description***IRQ109 level***Type***RW***Reset value***0***IRQ110****Location***27:24***Description***IRQ110 level***Type***RW*

Reset value

0

IRQ111**Location**

31:28

Description*IRQ111 level***Type***RW***Reset value**

0

5.24. qc.mclcivl14

IRQ Level 14

Level bits for IRQs 112-119

5.24.1. Attributes

Defining Extension	- anyOf: - Xqci, version >=0.7 - Xqciint, version >=0.4
CSR Address	0xbce
Length	32-bit
Privilege Mode	M

5.24.2. Format

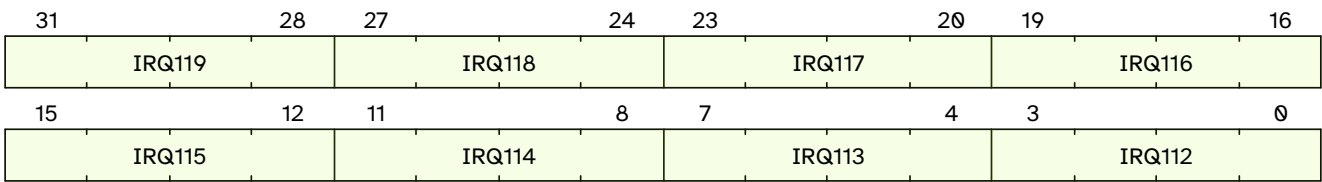


Figure 24. qc.mclcivl14 format

5.24.3. Field Summary

Name	Location	Type	Reset Value
qc.mclcivl14.IRQ112	3:0	RW	0
qc.mclcivl14.IRQ113	7:4	RW	0
qc.mclcivl14.IRQ114	11:8	RW	0
qc.mclcivl14.IRQ115	15:12	RW	0
qc.mclcivl14.IRQ116	19:16	RW	0
qc.mclcivl14.IRQ117	23:20	RW	0
qc.mclcivl14.IRQ118	27:24	RW	0
qc.mclcivl14.IRQ119	31:28	RW	0

5.24.4. Fields

IRQ112

Location
3:0

Description*IRQ112 level***Type***RW***Reset value***0***IRQ113****Location***7:4***Description***IRQ113 level***Type***RW***Reset value***0***IRQ114****Location***11:8***Description***IRQ114 level***Type***RW***Reset value***0***IRQ115****Location***15:12***Description***IRQ115 level*

Type*RW***Reset value***0***IRQ116****Location***19:16***Description***IRQ116 level***Type***RW***Reset value***0***IRQ117****Location***23:20***Description***IRQ117 level***Type***RW***Reset value***0***IRQ118****Location***27:24***Description***IRQ118 level***Type***RW*

Reset value

0

IRQ119**Location**

31:28

Description*IRQ119 level***Type***RW***Reset value**

0

5.25. qc.mclcivl15

IRQ Level 15

Level bits for IRQs 120-127

5.25.1. Attributes

Defining Extension	- anyOf: - Xqci, version >=0.7 - Xqciint, version >=0.4
CSR Address	0xbcf
Length	32-bit
Privilege Mode	M

5.25.2. Format

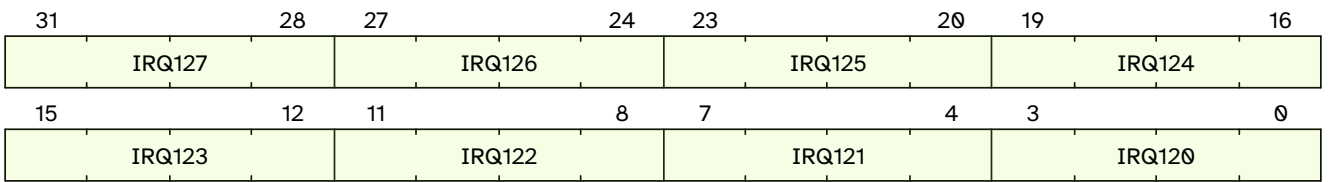


Figure 25. qc.mclcivl15 format

5.25.3. Field Summary

Name	Location	Type	Reset Value
qc.mclcivl15.IRQ120	3:0	RW	0
qc.mclcivl15.IRQ121	7:4	RW	0
qc.mclcivl15.IRQ122	11:8	RW	0
qc.mclcivl15.IRQ123	15:12	RW	0
qc.mclcivl15.IRQ124	19:16	RW	0
qc.mclcivl15.IRQ125	23:20	RW	0
qc.mclcivl15.IRQ126	27:24	RW	0
qc.mclcivl15.IRQ127	31:28	RW	0

5.25.4. Fields

IRQ120

Location
3:0

Description*IRQ120 level***Type***RW***Reset value***0***IRQ121****Location***7:4***Description***IRQ121 level***Type***RW***Reset value***0***IRQ122****Location***11:8***Description***IRQ122 level***Type***RW***Reset value***0***IRQ123****Location***15:12***Description***IRQ123 level*

Type*RW***Reset value***0***IRQ124****Location***19:16***Description***IRQ124 level***Type***RW***Reset value***0***IRQ125****Location***23:20***Description***IRQ125 level***Type***RW***Reset value***0***IRQ126****Location***27:24***Description***IRQ126 level***Type***RW*

Reset value*0***IRQ127****Location***31:28***Description***IRQ127 level***Type***RW***Reset value***0*

5.26. qc.mclivil16

IRQ Level 16

Level bits for IRQs 128-135

5.26.1. Attributes

Defining Extension	- anyOf: - Xqci, version >=0.7 - Xqciint, version >=0.4
CSR Address	0xbd0
Length	32-bit
Privilege Mode	M

5.26.2. Format

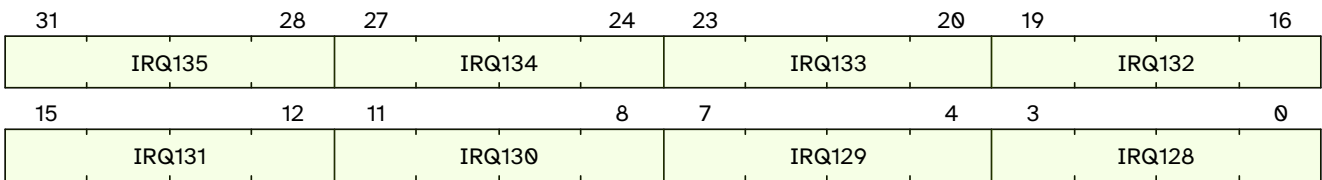


Figure 26. qc.mclivil16 format

5.26.3. Field Summary

Name	Location	Type	Reset Value
qc.mclivil16.IRQ128	3:0	RW	0
qc.mclivil16.IRQ129	7:4	RW	0
qc.mclivil16.IRQ130	11:8	RW	0
qc.mclivil16.IRQ131	15:12	RW	0
qc.mclivil16.IRQ132	19:16	RW	0
qc.mclivil16.IRQ133	23:20	RW	0
qc.mclivil16.IRQ134	27:24	RW	0
qc.mclivil16.IRQ135	31:28	RW	0

5.26.4. Fields

IRQ128

Location

3:0

Description*IRQ128 level***Type***RW***Reset value***0***IRQ129****Location***7:4***Description***IRQ129 level***Type***RW***Reset value***0***IRQ130****Location***11:8***Description***IRQ130 level***Type***RW***Reset value***0***IRQ131****Location***15:12***Description***IRQ131 level*

Type*RW***Reset value***0***IRQ132****Location***19:16***Description***IRQ132 level***Type***RW***Reset value***0***IRQ133****Location***23:20***Description***IRQ133 level***Type***RW***Reset value***0***IRQ134****Location***27:24***Description***IRQ134 level***Type***RW*

Reset value

0

IRQ135**Location**

31:28

Description*IRQ135 level***Type***RW***Reset value**

0

5.27. qc.mclcivl17

IRQ Level 17

Level bits for IRQs 136-143

5.27.1. Attributes

Defining Extension	- anyOf: - Xqci, version >=0.7 - Xqciint, version >=0.4
CSR Address	0xbd1
Length	32-bit
Privilege Mode	M

5.27.2. Format

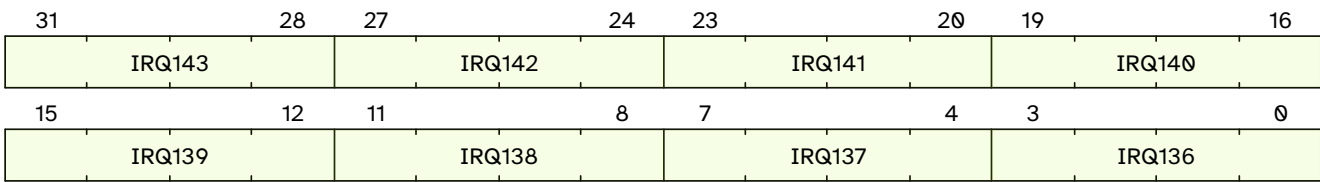


Figure 27. qc.mclcivl17 format

5.27.3. Field Summary

Name	Location	Type	Reset Value
qc.mclcivl17.IRQ136	3:0	RW	0
qc.mclcivl17.IRQ137	7:4	RW	0
qc.mclcivl17.IRQ138	11:8	RW	0
qc.mclcivl17.IRQ139	15:12	RW	0
qc.mclcivl17.IRQ140	19:16	RW	0
qc.mclcivl17.IRQ141	23:20	RW	0
qc.mclcivl17.IRQ142	27:24	RW	0
qc.mclcivl17.IRQ143	31:28	RW	0

5.27.4. Fields

IRQ136

Location
3:0

Description*IRQ136 level***Type***RW***Reset value***0***IRQ137****Location***7:4***Description***IRQ137 level***Type***RW***Reset value***0***IRQ138****Location***11:8***Description***IRQ138 level***Type***RW***Reset value***0***IRQ139****Location***15:12***Description***IRQ139 level*

Type*RW***Reset value**

0

IRQ140**Location***19:16***Description***IRQ140 level***Type***RW***Reset value**

0

IRQ141**Location***23:20***Description***IRQ141 level***Type***RW***Reset value**

0

IRQ142**Location***27:24***Description***IRQ142 level***Type***RW*

Reset value*0***IRQ143****Location***31:28***Description***IRQ143 level***Type***RW***Reset value***0*

5.28. qc.mclcivl18

IRQ Level 18

Level bits for IRQs 144-151

5.28.1. Attributes

Defining Extension	- anyOf: - Xqci, version >=0.7 - Xqciint, version >=0.4
CSR Address	0xbd2
Length	32-bit
Privilege Mode	M

5.28.2. Format

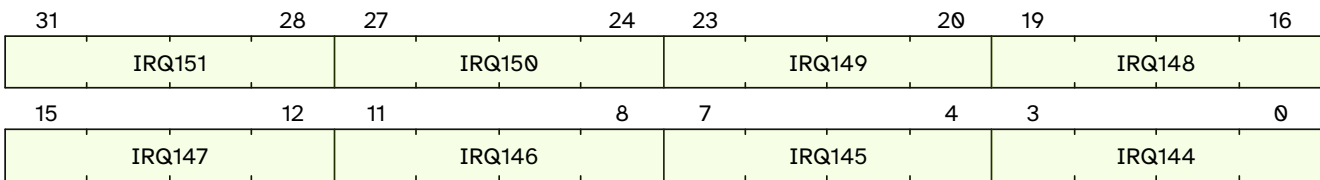


Figure 28. qc.mclcivl18 format

5.28.3. Field Summary

Name	Location	Type	Reset Value
qc.mclcivl18.IRQ144	3:0	RW	0
qc.mclcivl18.IRQ145	7:4	RW	0
qc.mclcivl18.IRQ146	11:8	RW	0
qc.mclcivl18.IRQ147	15:12	RW	0
qc.mclcivl18.IRQ148	19:16	RW	0
qc.mclcivl18.IRQ149	23:20	RW	0
qc.mclcivl18.IRQ150	27:24	RW	0
qc.mclcivl18.IRQ151	31:28	RW	0

5.28.4. Fields

IRQ144

Location

3:0

Description*IRQ144 level***Type***RW***Reset value***0***IRQ145****Location***7:4***Description***IRQ145 level***Type***RW***Reset value***0***IRQ146****Location***11:8***Description***IRQ146 level***Type***RW***Reset value***0***IRQ147****Location***15:12***Description***IRQ147 level*

Type*RW***Reset value***0***IRQ148****Location***19:16***Description***IRQ148 level***Type***RW***Reset value***0***IRQ149****Location***23:20***Description***IRQ149 level***Type***RW***Reset value***0***IRQ150****Location***27:24***Description***IRQ150 level***Type***RW*

Reset value

0

IRQ151

Location

31:28

Description

IRQ151 level

Type

RW

Reset value

0

5.29. qc.mclcivl19

IRQ Level 19

Level bits for IRQs 152-159

5.29.1. Attributes

Defining Extension	- anyOf: - Xqci, version >=0.7 - Xqciint, version >=0.4
CSR Address	0xbd3
Length	32-bit
Privilege Mode	M

5.29.2. Format

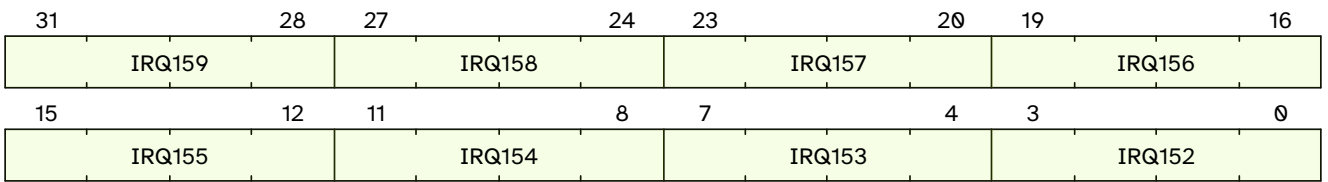


Figure 29. qc.mclcivl19 format

5.29.3. Field Summary

Name	Location	Type	Reset Value
qc.mclcivl19.IRQ152	3:0	RW	0
qc.mclcivl19.IRQ153	7:4	RW	0
qc.mclcivl19.IRQ154	11:8	RW	0
qc.mclcivl19.IRQ155	15:12	RW	0
qc.mclcivl19.IRQ156	19:16	RW	0
qc.mclcivl19.IRQ157	23:20	RW	0
qc.mclcivl19.IRQ158	27:24	RW	0
qc.mclcivl19.IRQ159	31:28	RW	0

5.29.4. Fields

IRQ152

Location
3:0

Description*IRQ152 level***Type***RW***Reset value***0***IRQ153****Location***7:4***Description***IRQ153 level***Type***RW***Reset value***0***IRQ154****Location***11:8***Description***IRQ154 level***Type***RW***Reset value***0***IRQ155****Location***15:12***Description***IRQ155 level*

Type*RW***Reset value***0***IRQ156****Location***19:16***Description***IRQ156 level***Type***RW***Reset value***0***IRQ157****Location***23:20***Description***IRQ157 level***Type***RW***Reset value***0***IRQ158****Location***27:24***Description***IRQ158 level***Type***RW*

Reset value*0***IRQ159****Location***31:28***Description***IRQ159 level***Type***RW***Reset value***0*

5.30. qc.mclcivl20

IRQ Level 20

Level bits for IRQs 160-167

5.30.1. Attributes

Defining Extension	- anyOf: - Xqci, version >=0.7 - Xqciint, version >=0.4
CSR Address	0xbd4
Length	32-bit
Privilege Mode	M

5.30.2. Format

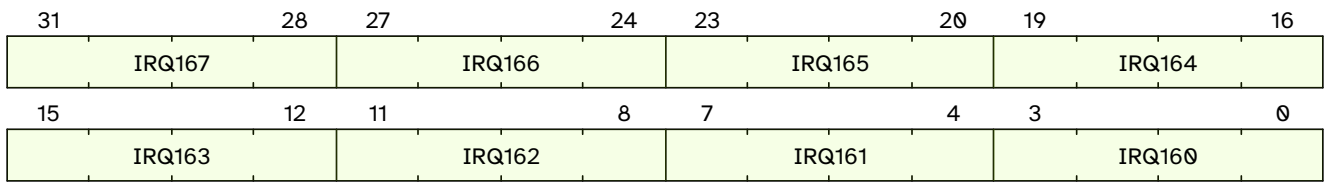


Figure 30. qc.mclcivl20 format

5.30.3. Field Summary

Name	Location	Type	Reset Value
qc.mclcivl20.IRQ160	3:0	RW	0
qc.mclcivl20.IRQ161	7:4	RW	0
qc.mclcivl20.IRQ162	11:8	RW	0
qc.mclcivl20.IRQ163	15:12	RW	0
qc.mclcivl20.IRQ164	19:16	RW	0
qc.mclcivl20.IRQ165	23:20	RW	0
qc.mclcivl20.IRQ166	27:24	RW	0
qc.mclcivl20.IRQ167	31:28	RW	0

5.30.4. Fields

IRQ160

Location
3:0

Description*IRQ160 level***Type***RW***Reset value***0***IRQ161****Location***7:4***Description***IRQ161 level***Type***RW***Reset value***0***IRQ162****Location***11:8***Description***IRQ162 level***Type***RW***Reset value***0***IRQ163****Location***15:12***Description***IRQ163 level*

Type*RW***Reset value***0***IRQ164****Location***19:16***Description***IRQ164 level***Type***RW***Reset value***0***IRQ165****Location***23:20***Description***IRQ165 level***Type***RW***Reset value***0***IRQ166****Location***27:24***Description***IRQ166 level***Type***RW*

Reset value

0

IRQ167

Location

31:28

Description

IRQ167 level

Type

RW

Reset value

0

5.31. qc.mclcivl21

IRQ Level 21

Level bits for IRQs 168-175

5.31.1. Attributes

Defining Extension	- anyOf: - Xqci, version >=0.7 - Xqciint, version >=0.4
CSR Address	0xbd5
Length	32-bit
Privilege Mode	M

5.31.2. Format

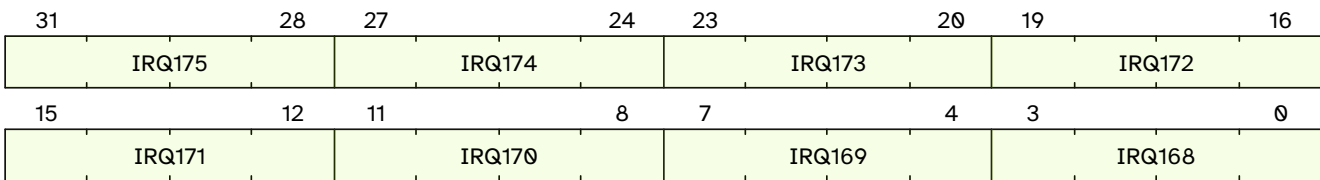


Figure 31. qc.mclcivl21 format

5.31.3. Field Summary

Name	Location	Type	Reset Value
qc.mclcivl21.IRQ168	3:0	RW	0
qc.mclcivl21.IRQ169	7:4	RW	0
qc.mclcivl21.IRQ170	11:8	RW	0
qc.mclcivl21.IRQ171	15:12	RW	0
qc.mclcivl21.IRQ172	19:16	RW	0
qc.mclcivl21.IRQ173	23:20	RW	0
qc.mclcivl21.IRQ174	27:24	RW	0
qc.mclcivl21.IRQ175	31:28	RW	0

5.31.4. Fields

IRQ168

Location

3:0

Description*IRQ168 level***Type***RW***Reset value***0***IRQ169****Location***7:4***Description***IRQ169 level***Type***RW***Reset value***0***IRQ170****Location***11:8***Description***IRQ170 level***Type***RW***Reset value***0***IRQ171****Location***15:12***Description***IRQ171 level*

Type*RW***Reset value***0***IRQ172****Location***19:16***Description***IRQ172 level***Type***RW***Reset value***0***IRQ173****Location***23:20***Description***IRQ173 level***Type***RW***Reset value***0***IRQ174****Location***27:24***Description***IRQ174 level***Type***RW*

Reset value 0

IRQ175

Location 31:28
Description IRQ175 level
Type RW
Reset value 0

5.32. qc.mclcivl22

IRQ Level 22

Level bits for IRQs 176-183

5.32.1. Attributes

Defining Extension	- anyOf: - Xqci, version >=0.7 - Xqciint, version >=0.4
CSR Address	0xbd6
Length	32-bit
Privilege Mode	M

5.32.2. Format

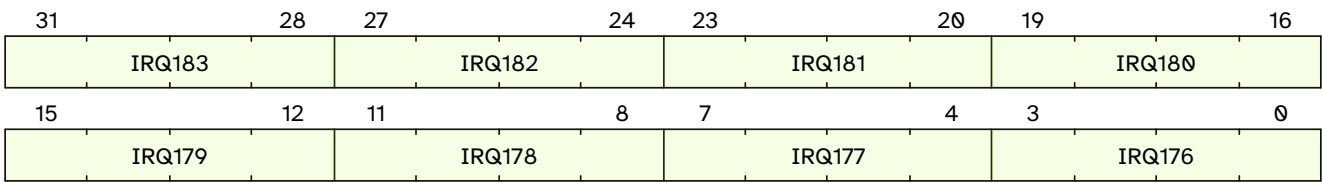


Figure 32. qc.mclcivl22 format

5.32.3. Field Summary

Name	Location	Type	Reset Value
qc.mclcivl22.IRQ176	3:0	RW	0
qc.mclcivl22.IRQ177	7:4	RW	0
qc.mclcivl22.IRQ178	11:8	RW	0
qc.mclcivl22.IRQ179	15:12	RW	0
qc.mclcivl22.IRQ180	19:16	RW	0
qc.mclcivl22.IRQ181	23:20	RW	0
qc.mclcivl22.IRQ182	27:24	RW	0
qc.mclcivl22.IRQ183	31:28	RW	0

5.32.4. Fields

IRQ176

Location

3:0

Description*IRQ176 level***Type***RW***Reset value***0***IRQ177****Location***7:4***Description***IRQ177 level***Type***RW***Reset value***0***IRQ178****Location***11:8***Description***IRQ178 level***Type***RW***Reset value***0***IRQ179****Location***15:12***Description***IRQ179 level*

Type*RW***Reset value***0***IRQ180****Location***19:16***Description***IRQ180 level***Type***RW***Reset value***0***IRQ181****Location***23:20***Description***IRQ181 level***Type***RW***Reset value***0***IRQ182****Location***27:24***Description***IRQ182 level***Type***RW*

Reset value

0

IRQ183

Location

31:28

Description

IRQ183 level

Type

RW

Reset value

0

5.33. qc.mclcivl23

IRQ Level 23

Level bits for IRQs 184-191

5.33.1. Attributes

Defining Extension	- anyOf: - Xqci, version >=0.7 - Xqciint, version >=0.4
CSR Address	0xbd7
Length	32-bit
Privilege Mode	M

5.33.2. Format

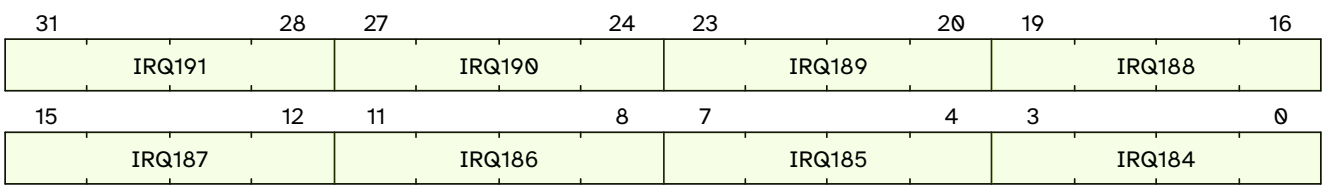


Figure 33. qc.mclcivl23 format

5.33.3. Field Summary

Name	Location	Type	Reset Value
qc.mclcivl23.IRQ184	3:0	RW	0
qc.mclcivl23.IRQ185	7:4	RW	0
qc.mclcivl23.IRQ186	11:8	RW	0
qc.mclcivl23.IRQ187	15:12	RW	0
qc.mclcivl23.IRQ188	19:16	RW	0
qc.mclcivl23.IRQ189	23:20	RW	0
qc.mclcivl23.IRQ190	27:24	RW	0
qc.mclcivl23.IRQ191	31:28	RW	0

5.33.4. Fields

IRQ184

Location

3:0

Description*IRQ184 level***Type***RW***Reset value***0***IRQ185****Location***7:4***Description***IRQ185 level***Type***RW***Reset value***0***IRQ186****Location***11:8***Description***IRQ186 level***Type***RW***Reset value***0***IRQ187****Location***15:12***Description***IRQ187 level*

Type*RW***Reset value***0***IRQ188****Location***19:16***Description***IRQ188 level***Type***RW***Reset value***0***IRQ189****Location***23:20***Description***IRQ189 level***Type***RW***Reset value***0***IRQ190****Location***27:24***Description***IRQ190 level***Type***RW*

Reset value

0

IRQ191

Location

31:28

Description

IRQ191 level

Type

RW

Reset value

0

5.34. qc.mclcivl24

IRQ Level 24

Level bits for IRQs 192-199

5.34.1. Attributes

Defining Extension	- anyOf: - Xqci, version >=0.7 - Xqciint, version >=0.4
CSR Address	0xbd8
Length	32-bit
Privilege Mode	M

5.34.2. Format

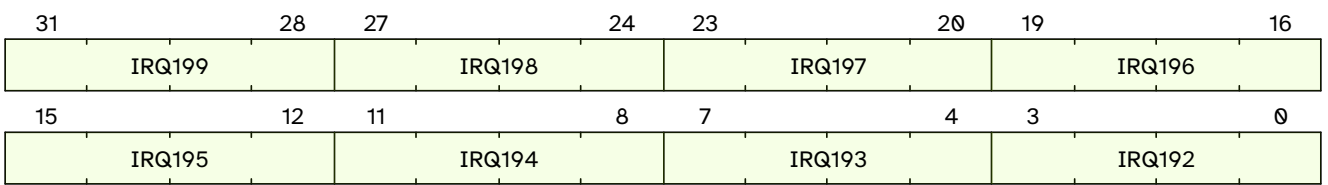


Figure 34. qc.mclcivl24 format

5.34.3. Field Summary

Name	Location	Type	Reset Value
qc.mclcivl24.IRQ192	3:0	RW	0
qc.mclcivl24.IRQ193	7:4	RW	0
qc.mclcivl24.IRQ194	11:8	RW	0
qc.mclcivl24.IRQ195	15:12	RW	0
qc.mclcivl24.IRQ196	19:16	RW	0
qc.mclcivl24.IRQ197	23:20	RW	0
qc.mclcivl24.IRQ198	27:24	RW	0
qc.mclcivl24.IRQ199	31:28	RW	0

5.34.4. Fields

IRQ192

Location
3:0

Description*IRQ192 level***Type***RW***Reset value***0***IRQ193****Location***7:4***Description***IRQ193 level***Type***RW***Reset value***0***IRQ194****Location***11:8***Description***IRQ194 level***Type***RW***Reset value***0***IRQ195****Location***15:12***Description***IRQ195 level*

Type*RW***Reset value***0***IRQ196****Location***19:16***Description***IRQ196 level***Type***RW***Reset value***0***IRQ197****Location***23:20***Description***IRQ197 level***Type***RW***Reset value***0***IRQ198****Location***27:24***Description***IRQ198 level***Type***RW*

Reset value

0

IRQ199

Location

31:28

Description

IRQ199 level

Type

RW

Reset value

0

5.35. qc.mclcivl25

IRQ Level 25

Level bits for IRQs 200-207

5.35.1. Attributes

Defining Extension	- anyOf: - Xqci, version >=0.7 - Xqciint, version >=0.4
CSR Address	0xbd9
Length	32-bit
Privilege Mode	M

5.35.2. Format

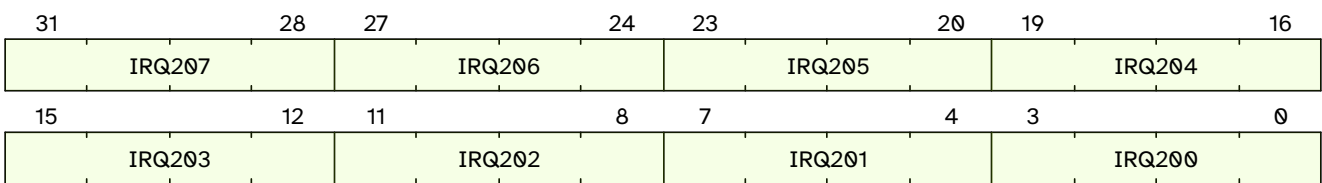


Figure 35. qc.mclcivl25 format

5.35.3. Field Summary

Name	Location	Type	Reset Value
qc.mclcivl25.IRQ200	3:0	RW	0
qc.mclcivl25.IRQ201	7:4	RW	0
qc.mclcivl25.IRQ202	11:8	RW	0
qc.mclcivl25.IRQ203	15:12	RW	0
qc.mclcivl25.IRQ204	19:16	RW	0
qc.mclcivl25.IRQ205	23:20	RW	0
qc.mclcivl25.IRQ206	27:24	RW	0
qc.mclcivl25.IRQ207	31:28	RW	0

5.35.4. Fields

IRQ200

Location
3:0

Description*IRQ200 level***Type***RW***Reset value***0***IRQ201****Location***7:4***Description***IRQ201 level***Type***RW***Reset value***0***IRQ202****Location***11:8***Description***IRQ202 level***Type***RW***Reset value***0***IRQ203****Location***15:12***Description***IRQ203 level*

Type*RW***Reset value***0***IRQ204****Location***19:16***Description***IRQ204 level***Type***RW***Reset value***0***IRQ205****Location***23:20***Description***IRQ205 level***Type***RW***Reset value***0***IRQ206****Location***27:24***Description***IRQ206 level***Type***RW*

Reset value

0

IRQ207

Location

31:28

Description

IRQ207 level

Type

RW

Reset value

0

5.36. qc.mclcivl26

IRQ Level 26

Level bits for IRQs 208-215

5.36.1. Attributes

Defining Extension	- anyOf: - Xqci, version >=0.7 - Xqciint, version >=0.4
CSR Address	0xbda
Length	32-bit
Privilege Mode	M

5.36.2. Format

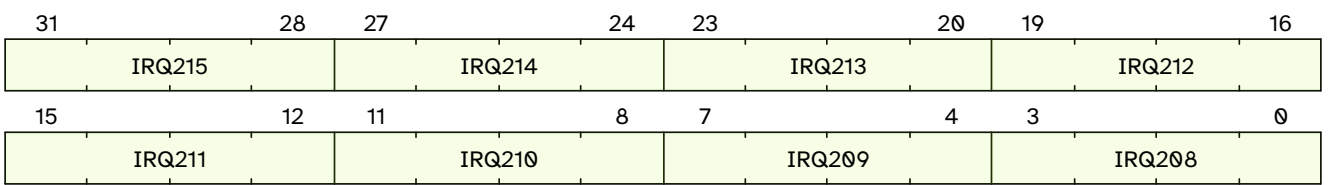


Figure 36. qc.mclcivl26 format

5.36.3. Field Summary

Name	Location	Type	Reset Value
qc.mclcivl26.IRQ208	3:0	RW	0
qc.mclcivl26.IRQ209	7:4	RW	0
qc.mclcivl26.IRQ210	11:8	RW	0
qc.mclcivl26.IRQ211	15:12	RW	0
qc.mclcivl26.IRQ212	19:16	RW	0
qc.mclcivl26.IRQ213	23:20	RW	0
qc.mclcivl26.IRQ214	27:24	RW	0
qc.mclcivl26.IRQ215	31:28	RW	0

5.36.4. Fields

IRQ208

Location
3:0

Description*IRQ208 level***Type***RW***Reset value***0***IRQ209****Location***7:4***Description***IRQ209 level***Type***RW***Reset value***0***IRQ210****Location***11:8***Description***IRQ210 level***Type***RW***Reset value***0***IRQ211****Location***15:12***Description***IRQ211 level*

Type*RW***Reset value**

0

IRQ212**Location***19:16***Description***IRQ212 level***Type***RW***Reset value**

0

IRQ213**Location***23:20***Description***IRQ213 level***Type***RW***Reset value**

0

IRQ214**Location***27:24***Description***IRQ214 level***Type***RW*

Reset value

0

IRQ215

Location

31:28

Description

IRQ215 level

Type

RW

Reset value

0

5.37. qc.mclcivl27

IRQ Level 27

Level bits for IRQs 216-223

5.37.1. Attributes

Defining Extension	- anyOf: - Xqci, version >=0.7 - Xqciint, version >=0.4
CSR Address	0xbdb
Length	32-bit
Privilege Mode	M

5.37.2. Format

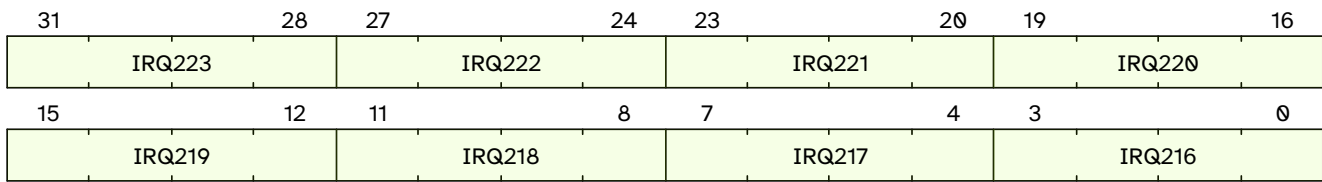


Figure 37. qc.mclcivl27 format

5.37.3. Field Summary

Name	Location	Type	Reset Value
qc.mclcivl27.IRQ216	3:0	RW	0
qc.mclcivl27.IRQ217	7:4	RW	0
qc.mclcivl27.IRQ218	11:8	RW	0
qc.mclcivl27.IRQ219	15:12	RW	0
qc.mclcivl27.IRQ220	19:16	RW	0
qc.mclcivl27.IRQ221	23:20	RW	0
qc.mclcivl27.IRQ222	27:24	RW	0
qc.mclcivl27.IRQ223	31:28	RW	0

5.37.4. Fields

IRQ216

Location
3:0

Description*IRQ216 level***Type***RW***Reset value***0***IRQ217****Location***7:4***Description***IRQ217 level***Type***RW***Reset value***0***IRQ218****Location***11:8***Description***IRQ218 level***Type***RW***Reset value***0***IRQ219****Location***15:12***Description***IRQ219 level*

Type*RW***Reset value***0***IRQ220****Location***19:16***Description***IRQ220 level***Type***RW***Reset value***0***IRQ221****Location***23:20***Description***IRQ221 level***Type***RW***Reset value***0***IRQ222****Location***27:24***Description***IRQ222 level***Type***RW*

Reset value

0

IRQ223

Location

31:28

Description

IRQ223 level

Type

RW

Reset value

0

5.38. qc.mclcivl28

IRQ Level 28

Level bits for IRQs 224-231

5.38.1. Attributes

Defining Extension	- anyOf: - Xqci, version >=0.7 - Xqciint, version >=0.4
CSR Address	0xbdc
Length	32-bit
Privilege Mode	M

5.38.2. Format

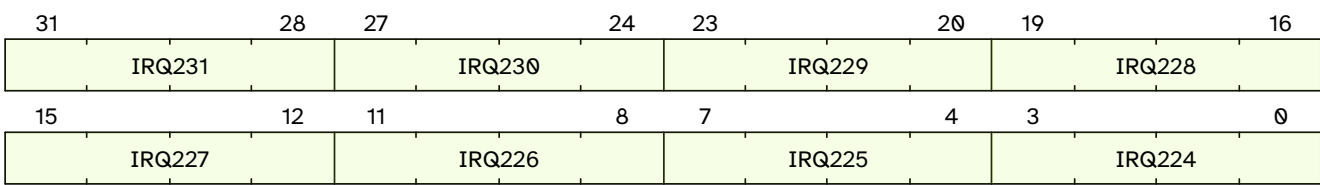


Figure 38. qc.mclcivl28 format

5.38.3. Field Summary

Name	Location	Type	Reset Value
qc.mclcivl28.IRQ224	3:0	RW	0
qc.mclcivl28.IRQ225	7:4	RW	0
qc.mclcivl28.IRQ226	11:8	RW	0
qc.mclcivl28.IRQ227	15:12	RW	0
qc.mclcivl28.IRQ228	19:16	RW	0
qc.mclcivl28.IRQ229	23:20	RW	0
qc.mclcivl28.IRQ230	27:24	RW	0
qc.mclcivl28.IRQ231	31:28	RW	0

5.38.4. Fields

IRQ224

Location
3:0

Description*IRQ224 level***Type***RW***Reset value***0***IRQ225****Location***7:4***Description***IRQ225 level***Type***RW***Reset value***0***IRQ226****Location***11:8***Description***IRQ226 level***Type***RW***Reset value***0***IRQ227****Location***15:12***Description***IRQ227 level*

Type*RW***Reset value***0***IRQ228****Location***19:16***Description***IRQ228 level***Type***RW***Reset value***0***IRQ229****Location***23:20***Description***IRQ229 level***Type***RW***Reset value***0***IRQ230****Location***27:24***Description***IRQ230 level***Type***RW*

Reset value

0

IRQ231

Location

31:28

Description

IRQ231 level

Type

RW

Reset value

0

5.39. qc.mclivil29

IRQ Level 29

Level bits for IRQs 232-239

5.39.1. Attributes

Defining Extension	- anyOf: - Xqci, version >=0.7 - Xqciint, version >=0.4
CSR Address	0xbdd
Length	32-bit
Privilege Mode	M

5.39.2. Format

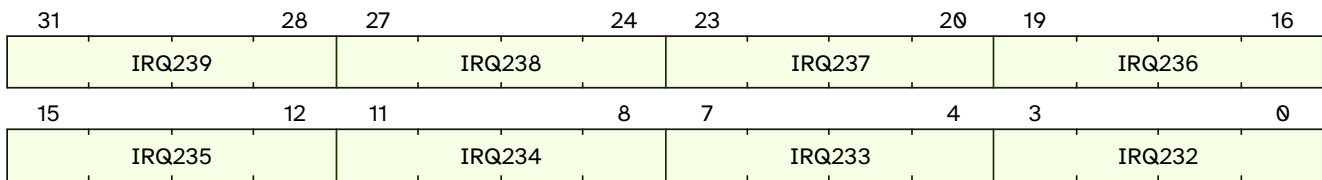


Figure 39. qc.mclivil29 format

5.39.3. Field Summary

Name	Location	Type	Reset Value
qc.mclivil29.IRQ232	3:0	RW	0
qc.mclivil29.IRQ233	7:4	RW	0
qc.mclivil29.IRQ234	11:8	RW	0
qc.mclivil29.IRQ235	15:12	RW	0
qc.mclivil29.IRQ236	19:16	RW	0
qc.mclivil29.IRQ237	23:20	RW	0
qc.mclivil29.IRQ238	27:24	RW	0
qc.mclivil29.IRQ239	31:28	RW	0

5.39.4. Fields

IRQ232

Location
3:0

Description*IRQ232 level***Type***RW***Reset value**

0

IRQ233**Location***7:4***Description***IRQ233 level***Type***RW***Reset value**

0

IRQ234**Location***11:8***Description***IRQ234 level***Type***RW***Reset value**

0

IRQ235**Location***15:12***Description***IRQ235 level*

Type*RW***Reset value***0***IRQ236****Location***19:16***Description***IRQ236 level***Type***RW***Reset value***0***IRQ237****Location***23:20***Description***IRQ237 level***Type***RW***Reset value***0***IRQ238****Location***27:24***Description***IRQ238 level***Type***RW*

Reset value

0

IRQ239

Location

31:28

Description

IRQ239 level

Type

RW

Reset value

0

5.40. qc.mclivil30

IRQ Level 30

Level bits for IRQs 240-247

5.40.1. Attributes

Defining Extension	- anyOf: - Xqci, version >=0.7 - Xqciint, version >=0.4
CSR Address	0xbde
Length	32-bit
Privilege Mode	M

5.40.2. Format

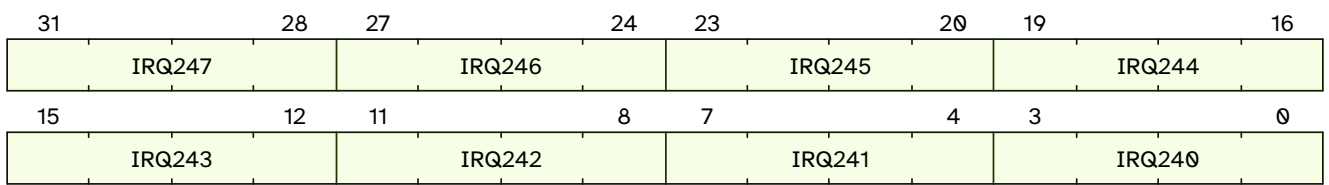


Figure 40. qc.mclivil30 format

5.40.3. Field Summary

Name	Location	Type	Reset Value
qc.mclivil30.IRQ240	3:0	RW	0
qc.mclivil30.IRQ241	7:4	RW	0
qc.mclivil30.IRQ242	11:8	RW	0
qc.mclivil30.IRQ243	15:12	RW	0
qc.mclivil30.IRQ244	19:16	RW	0
qc.mclivil30.IRQ245	23:20	RW	0
qc.mclivil30.IRQ246	27:24	RW	0
qc.mclivil30.IRQ247	31:28	RW	0

5.40.4. Fields

IRQ240

Location
3:0

Description*IRQ240 level***Type***RW***Reset value***0***IRQ241****Location***7:4***Description***IRQ241 level***Type***RW***Reset value***0***IRQ242****Location***11:8***Description***IRQ242 level***Type***RW***Reset value***0***IRQ243****Location***15:12***Description***IRQ243 level*

Type*RW***Reset value***0***IRQ244****Location***19:16***Description***IRQ244 level***Type***RW***Reset value***0***IRQ245****Location***23:20***Description***IRQ245 level***Type***RW***Reset value***0***IRQ246****Location***27:24***Description***IRQ246 level***Type***RW*

Reset value

0

IRQ247

Location

31:28

Description

IRQ247 level

Type

RW

Reset value

0

5.41. qc.mclcivl31

IRQ Level 31

Level bits for IRQs 248-255

5.41.1. Attributes

Defining Extension	- anyOf: - Xqci, version >=0.7 - Xqciint, version >=0.4
CSR Address	0xbdf
Length	32-bit
Privilege Mode	M

5.41.2. Format

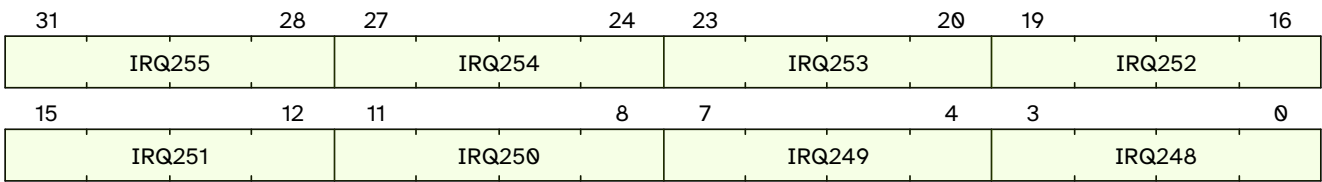


Figure 41. qc.mclcivl31 format

5.41.3. Field Summary

Name	Location	Type	Reset Value
qc.mclcivl31.IRQ248	3:0	RW	0
qc.mclcivl31.IRQ249	7:4	RW	0
qc.mclcivl31.IRQ250	11:8	RW	0
qc.mclcivl31.IRQ251	15:12	RW	0
qc.mclcivl31.IRQ252	19:16	RW	0
qc.mclcivl31.IRQ253	23:20	RW	0
qc.mclcivl31.IRQ254	27:24	RW	0
qc.mclcivl31.IRQ255	31:28	RW	0

5.41.4. Fields

IRQ248

Location

3:0

Description*IRQ248 level***Type***RW***Reset value***0***IRQ249****Location***7:4***Description***IRQ249 level***Type***RW***Reset value***0***IRQ250****Location***11:8***Description***IRQ250 level***Type***RW***Reset value***0***IRQ251****Location***15:12***Description***IRQ251 level*

Type*RW***Reset value**

0

IRQ252**Location***19:16***Description***IRQ252 level***Type***RW***Reset value**

0

IRQ253**Location***23:20***Description***IRQ253 level***Type***RW***Reset value**

0

IRQ254**Location***27:24***Description***IRQ254 level***Type***RW*

Reset value

0

IRQ255**Location**

31:28

Description*IRQ255 level***Type***RW***Reset value**

0

5.42. qc.mclicip0

IRQ Pending 0

Pending bits for IRQs 0-31

5.42.1. Attributes

Defining Extension	- anyOf: - Xqci, version >= Xqci@0.1.0 - Xqciint, version >= Xqciint@0.1.0
CSR Address	0x7f0
Length	32-bit
Privilege Mode	M

5.42.2. Format

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
IRQ31	IRQ30	IRQ29	IRQ28	IRQ27	IRQ26	IRQ25	IRQ24	IRQ23	IRQ22	IRQ21	IRQ20	IRQ19	IRQ18	IRQ17	IRQ16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IRQ15	IRQ14	IRQ13	IRQ12	IRQ11	IRQ10	IRQ9	IRQ8	IRQ7	IRQ6	IRQ5	IRQ4	IRQ3	IRQ2	IRQ1	IRQ0

Figure 42. qc.mclicip0 format

5.42.3. Field Summary

Name	Location	Type	Reset Value
qc.mclicip0.IRQ0	0	RW	0
qc.mclicip0.IRQ1	1	RW	0
qc.mclicip0.IRQ2	2	RW	0
qc.mclicip0.IRQ3	3	RW	0
qc.mclicip0.IRQ4	4	RW	0
qc.mclicip0.IRQ5	5	RW	0
qc.mclicip0.IRQ6	6	RW	0
qc.mclicip0.IRQ7	7	RW	0
qc.mclicip0.IRQ8	8	RW	0
qc.mclicip0.IRQ9	9	RW	0
qc.mclicip0.IRQ10	10	RW	0
qc.mclicip0.IRQ11	11	RW	0
qc.mclicip0.IRQ12	12	RW	0
qc.mclicip0.IRQ13	13	RW	0
qc.mclicip0.IRQ14	14	RW	0

Name	Location	Type	Reset Value
qc.mclicip0.IRQ15	15	RW	0
qc.mclicip0.IRQ16	16	RW	0
qc.mclicip0.IRQ17	17	RW	0
qc.mclicip0.IRQ18	18	RW	0
qc.mclicip0.IRQ19	19	RW	0
qc.mclicip0.IRQ20	20	RW	0
qc.mclicip0.IRQ21	21	RW	0
qc.mclicip0.IRQ22	22	RW	0
qc.mclicip0.IRQ23	23	RW	0
qc.mclicip0.IRQ24	24	RW	0
qc.mclicip0.IRQ25	25	RW	0
qc.mclicip0.IRQ26	26	RW	0
qc.mclicip0.IRQ27	27	RW	0
qc.mclicip0.IRQ28	28	RW	0
qc.mclicip0.IRQ29	29	RW	0
qc.mclicip0.IRQ30	30	RW	0
qc.mclicip0.IRQ31	31	RW	0

5.42.4. Fields

IRQ0

Location

0

Description

IRQ0 pending

Type

RW

Reset value

0

IRQ1

Location

1

Description
<i>IRQ1 pending</i>
Type
<i>RW</i>
Reset value
0

IRQ2

Location
2
Description
<i>IRQ2 pending</i>
Type
<i>RW</i>
Reset value
0

IRQ3

Location
3
Description
<i>IRQ3 pending</i>
Type
<i>RW</i>
Reset value
0

IRQ4

Location
4
Description
<i>IRQ4 pending</i>

Type*RW***Reset value***0***IRQ5****Location***5***Description***IRQ5 pending***Type***RW***Reset value***0***IRQ6****Location***6***Description***IRQ6 pending***Type***RW***Reset value***0***IRQ7****Location***7***Description***IRQ7 pending***Type***RW*

Reset value
0

IRQ8

Location
8
Description
<i>IRQ8 pending</i>
Type
<i>RW</i>
Reset value
0

IRQ9

Location
9
Description
<i>IRQ9 pending</i>
Type
<i>RW</i>
Reset value
0

IRQ10

Location
10
Description
<i>IRQ10 pending</i>
Type
<i>RW</i>
Reset value
0

IRQ11**Location***11***Description***IRQ11 pending***Type***RW***Reset value***0***IRQ12****Location***12***Description***IRQ12 pending***Type***RW***Reset value***0***IRQ13****Location***13***Description***IRQ13 pending***Type***RW***Reset value***0*

IRQ14**Location***14***Description***IRQ14 pending***Type***RW***Reset value***0***IRQ15****Location***15***Description***IRQ15 pending***Type***RW***Reset value***0***IRQ16****Location***16***Description***IRQ16 pending***Type***RW***Reset value***0*

IRQ17**Location***17***Description***IRQ17 pending***Type***RW***Reset value***0***IRQ18****Location***18***Description***IRQ18 pending***Type***RW***Reset value***0***IRQ19****Location***19***Description***IRQ19 pending***Type***RW***Reset value***0*

IRQ20

Location
20
Description
<i>IRQ20 pending</i>
Type
<i>RW</i>
Reset value
0

IRQ21

Location
21
Description
<i>IRQ21 pending</i>
Type
<i>RW</i>
Reset value
0

IRQ22

Location
22
Description
<i>IRQ22 pending</i>
Type
<i>RW</i>
Reset value
0

IRQ23**Location**

23

Description*IRQ23 pending***Type***RW***Reset value**

0

IRQ24**Location**

24

Description*IRQ24 pending***Type***RW***Reset value**

0

IRQ25**Location**

25

Description*IRQ25 pending***Type***RW***Reset value**

0

IRQ26

Location

26

Description*IRQ26 pending***Type***RW***Reset value**

0

IRQ27

Location

27

Description*IRQ27 pending***Type***RW***Reset value**

0

IRQ28

Location

28

Description*IRQ28 pending***Type***RW***Reset value**

0

IRQ29

Location*29***Description***IRQ29 pending***Type***RW***Reset value***0*

IRQ30

Location*30***Description***IRQ30 pending***Type***RW***Reset value***0*

IRQ31

Location*31***Description***IRQ31 pending***Type***RW***Reset value***0*

5.43. qc.mclicip1

IRQ Pending 1

Pending bits for IRQs 32-63

5.43.1. Attributes

Defining Extension	- anyOf: - Xqci, version >= Xqci@0.1.0 - Xqciint, version >= Xqciint@0.1.0
CSR Address	0x7f1
Length	32-bit
Privilege Mode	M

5.43.2. Format

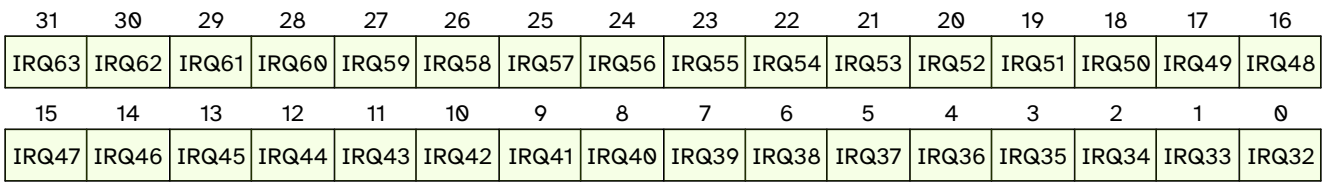


Figure 43. qc.mclicip1 format

5.43.3. Field Summary

Name	Location	Type	Reset Value
qc.mclicip1.IRQ32	0	RW	0
qc.mclicip1.IRQ33	1	RW	0
qc.mclicip1.IRQ34	2	RW	0
qc.mclicip1.IRQ35	3	RW	0
qc.mclicip1.IRQ36	4	RW	0
qc.mclicip1.IRQ37	5	RW	0
qc.mclicip1.IRQ38	6	RW	0
qc.mclicip1.IRQ39	7	RW	0
qc.mclicip1.IRQ40	8	RW	0
qc.mclicip1.IRQ41	9	RW	0
qc.mclicip1.IRQ42	10	RW	0
qc.mclicip1.IRQ43	11	RW	0
qc.mclicip1.IRQ44	12	RW	0
qc.mclicip1.IRQ45	13	RW	0
qc.mclicip1.IRQ46	14	RW	0

Name	Location	Type	Reset Value
qc.mclicip1.IRQ47	15	RW	0
qc.mclicip1.IRQ48	16	RW	0
qc.mclicip1.IRQ49	17	RW	0
qc.mclicip1.IRQ50	18	RW	0
qc.mclicip1.IRQ51	19	RW	0
qc.mclicip1.IRQ52	20	RW	0
qc.mclicip1.IRQ53	21	RW	0
qc.mclicip1.IRQ54	22	RW	0
qc.mclicip1.IRQ55	23	RW	0
qc.mclicip1.IRQ56	24	RW	0
qc.mclicip1.IRQ57	25	RW	0
qc.mclicip1.IRQ58	26	RW	0
qc.mclicip1.IRQ59	27	RW	0
qc.mclicip1.IRQ60	28	RW	0
qc.mclicip1.IRQ61	29	RW	0
qc.mclicip1.IRQ62	30	RW	0
qc.mclicip1.IRQ63	31	RW	0

5.43.4. Fields

IRQ32

Location

0

Description

IRQ32 pending

Type

RW

Reset value

0

IRQ33

Location

1

Description
<i>IRQ33 pending</i>
Type
<i>RW</i>
Reset value
<i>0</i>

IRQ34

Location
<i>2</i>
Description
<i>IRQ34 pending</i>
Type
<i>RW</i>
Reset value
<i>0</i>

IRQ35

Location
<i>3</i>
Description
<i>IRQ35 pending</i>
Type
<i>RW</i>
Reset value
<i>0</i>

IRQ36

Location
<i>4</i>
Description
<i>IRQ36 pending</i>

Type*RW***Reset value**

0

IRQ37**Location**

5

Description*IRQ37 pending***Type***RW***Reset value**

0

IRQ38**Location**

6

Description*IRQ38 pending***Type***RW***Reset value**

0

IRQ39**Location**

7

Description*IRQ39 pending***Type***RW*

Reset value

0

IRQ40**Location**

8

Description*IRQ40 pending***Type***RW***Reset value**

0

IRQ41**Location**

9

Description*IRQ41 pending***Type***RW***Reset value**

0

IRQ42**Location**

10

Description*IRQ42 pending***Type***RW***Reset value**

0

IRQ43**Location***11***Description***IRQ43 pending***Type***RW***Reset value***0***IRQ44****Location***12***Description***IRQ44 pending***Type***RW***Reset value***0***IRQ45****Location***13***Description***IRQ45 pending***Type***RW***Reset value***0*

IRQ46**Location***14***Description***IRQ46 pending***Type***RW***Reset value***0***IRQ47****Location***15***Description***IRQ47 pending***Type***RW***Reset value***0***IRQ48****Location***16***Description***IRQ48 pending***Type***RW***Reset value***0*

IRQ49

Location*17***Description***IRQ49 pending***Type***RW***Reset value***0*

IRQ50

Location*18***Description***IRQ50 pending***Type***RW***Reset value***0*

IRQ51

Location*19***Description***IRQ51 pending***Type***RW***Reset value***0*

IRQ52

Location

20

Description*IRQ52 pending***Type***RW***Reset value**

0

IRQ53

Location

21

Description*IRQ53 pending***Type***RW***Reset value**

0

IRQ54

Location

22

Description*IRQ54 pending***Type***RW***Reset value**

0

IRQ55**Location**

23

Description*IRQ55 pending***Type***RW***Reset value**

0

IRQ56**Location**

24

Description*IRQ56 pending***Type***RW***Reset value**

0

IRQ57**Location**

25

Description*IRQ57 pending***Type***RW***Reset value**

0

IRQ58

Location

26

Description*IRQ58 pending***Type***RW***Reset value**

0

IRQ59

Location

27

Description*IRQ59 pending***Type***RW***Reset value**

0

IRQ60

Location

28

Description*IRQ60 pending***Type***RW***Reset value**

0

IRQ61**Location***29***Description***IRQ61 pending***Type***RW***Reset value***0***IRQ62****Location***30***Description***IRQ62 pending***Type***RW***Reset value***0***IRQ63****Location***31***Description***IRQ63 pending***Type***RW***Reset value***0*

5.44. qc.mclicip2

IRQ Pending 2

Pending bits for IRQs 64-95

5.44.1. Attributes

Defining Extension	- anyOf: - Xqci, version >= Xqci@0.1.0 - Xqciint, version >= Xqciint@0.1.0
CSR Address	0x7f2
Length	32-bit
Privilege Mode	M

5.44.2. Format

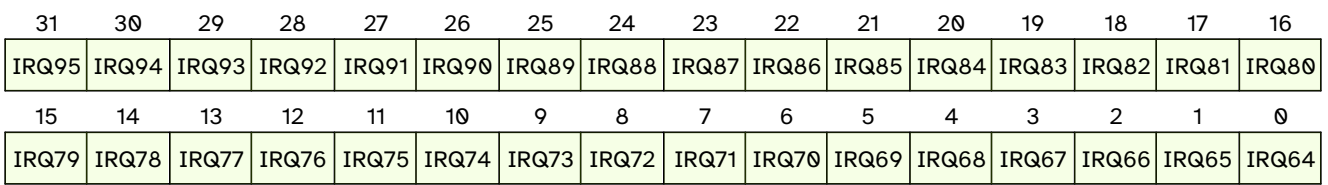


Figure 44. qc.mclicip2 format

5.44.3. Field Summary

Name	Location	Type	Reset Value
qc.mclicip2.IRQ64	0	RW	0
qc.mclicip2.IRQ65	1	RW	0
qc.mclicip2.IRQ66	2	RW	0
qc.mclicip2.IRQ67	3	RW	0
qc.mclicip2.IRQ68	4	RW	0
qc.mclicip2.IRQ69	5	RW	0
qc.mclicip2.IRQ70	6	RW	0
qc.mclicip2.IRQ71	7	RW	0
qc.mclicip2.IRQ72	8	RW	0
qc.mclicip2.IRQ73	9	RW	0
qc.mclicip2.IRQ74	10	RW	0
qc.mclicip2.IRQ75	11	RW	0
qc.mclicip2.IRQ76	12	RW	0
qc.mclicip2.IRQ77	13	RW	0
qc.mclicip2.IRQ78	14	RW	0

Name	Location	Type	Reset Value
qc.mclicip2.IRQ79	15	RW	0
qc.mclicip2.IRQ80	16	RW	0
qc.mclicip2.IRQ81	17	RW	0
qc.mclicip2.IRQ82	18	RW	0
qc.mclicip2.IRQ83	19	RW	0
qc.mclicip2.IRQ84	20	RW	0
qc.mclicip2.IRQ85	21	RW	0
qc.mclicip2.IRQ86	22	RW	0
qc.mclicip2.IRQ87	23	RW	0
qc.mclicip2.IRQ88	24	RW	0
qc.mclicip2.IRQ89	25	RW	0
qc.mclicip2.IRQ90	26	RW	0
qc.mclicip2.IRQ91	27	RW	0
qc.mclicip2.IRQ92	28	RW	0
qc.mclicip2.IRQ93	29	RW	0
qc.mclicip2.IRQ94	30	RW	0
qc.mclicip2.IRQ95	31	RW	0

5.44.4. Fields

IRQ64

Location

0

Description

IRQ64 pending

Type

RW

Reset value

0

IRQ65

Location

1

Description*IRQ65 pending***Type***RW***Reset value**

0

IRQ66**Location**

2

Description*IRQ66 pending***Type***RW***Reset value**

0

IRQ67**Location**

3

Description*IRQ67 pending***Type***RW***Reset value**

0

IRQ68**Location**

4

Description*IRQ68 pending*

Type*RW***Reset value**

0

IRQ69**Location**

5

Description*IRQ69 pending***Type***RW***Reset value**

0

IRQ70**Location**

6

Description*IRQ70 pending***Type***RW***Reset value**

0

IRQ71**Location**

7

Description*IRQ71 pending***Type***RW*

Reset value

0

IRQ72**Location**

8

Description*IRQ72 pending***Type***RW***Reset value**

0

IRQ73**Location**

9

Description*IRQ73 pending***Type***RW***Reset value**

0

IRQ74**Location**

10

Description*IRQ74 pending***Type***RW***Reset value**

0

IRQ75**Location***11***Description***IRQ75 pending***Type***RW***Reset value***0***IRQ76****Location***12***Description***IRQ76 pending***Type***RW***Reset value***0***IRQ77****Location***13***Description***IRQ77 pending***Type***RW***Reset value***0*

IRQ78**Location***14***Description***IRQ78 pending***Type***RW***Reset value***0***IRQ79****Location***15***Description***IRQ79 pending***Type***RW***Reset value***0***IRQ80****Location***16***Description***IRQ80 pending***Type***RW***Reset value***0*

IRQ81

Location*17***Description***IRQ81 pending***Type***RW***Reset value***0*

IRQ82

Location*18***Description***IRQ82 pending***Type***RW***Reset value***0*

IRQ83

Location*19***Description***IRQ83 pending***Type***RW***Reset value***0*

IRQ84

Location

20

Description*IRQ84 pending***Type***RW***Reset value**

0

IRQ85

Location

21

Description*IRQ85 pending***Type***RW***Reset value**

0

IRQ86

Location

22

Description*IRQ86 pending***Type***RW***Reset value**

0

IRQ87

Location

23

Description*IRQ87 pending***Type***RW***Reset value**

0

IRQ88

Location

24

Description*IRQ88 pending***Type***RW***Reset value**

0

IRQ89

Location

25

Description*IRQ89 pending***Type***RW***Reset value**

0

IRQ90**Location**

26

Description*IRQ90 pending***Type***RW***Reset value**

0

IRQ91**Location**

27

Description*IRQ91 pending***Type***RW***Reset value**

0

IRQ92**Location**

28

Description*IRQ92 pending***Type***RW***Reset value**

0

IRQ93**Location***29***Description***IRQ93 pending***Type***RW***Reset value***0***IRQ94****Location***30***Description***IRQ94 pending***Type***RW***Reset value***0***IRQ95****Location***31***Description***IRQ95 pending***Type***RW***Reset value***0*

5.45. qc.mclcip3

IRQ Pending 3

Pending bits for IRQs 96-127

5.45.1. Attributes

Defining Extension	- anyOf: - Xqci, version >= Xqci@0.1.0 - Xqciint, version >= Xqciint@0.1.0
CSR Address	0x7f3
Length	32-bit
Privilege Mode	M

5.45.2. Format

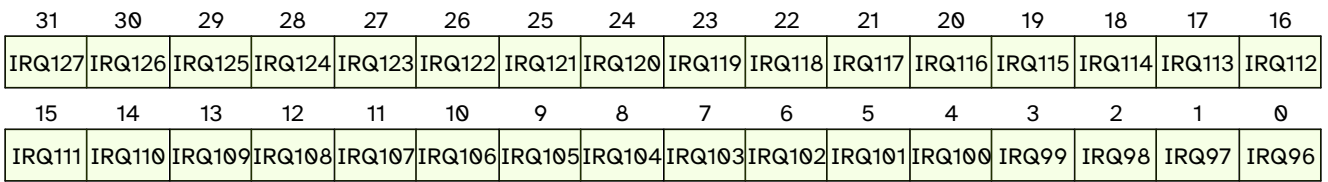


Figure 45. qc.mclcip3 format

5.45.3. Field Summary

Name	Location	Type	Reset Value
qc.mclcip3.IRQ96	0	RW	0
qc.mclcip3.IRQ97	1	RW	0
qc.mclcip3.IRQ98	2	RW	0
qc.mclcip3.IRQ99	3	RW	0
qc.mclcip3.IRQ100	4	RW	0
qc.mclcip3.IRQ101	5	RW	0
qc.mclcip3.IRQ102	6	RW	0
qc.mclcip3.IRQ103	7	RW	0
qc.mclcip3.IRQ104	8	RW	0
qc.mclcip3.IRQ105	9	RW	0
qc.mclcip3.IRQ106	10	RW	0
qc.mclcip3.IRQ107	11	RW	0
qc.mclcip3.IRQ108	12	RW	0
qc.mclcip3.IRQ109	13	RW	0
qc.mclcip3.IRQ110	14	RW	0

Name	Location	Type	Reset Value
qc.mclcip3.IRQ111	15	RW	0
qc.mclcip3.IRQ112	16	RW	0
qc.mclcip3.IRQ113	17	RW	0
qc.mclcip3.IRQ114	18	RW	0
qc.mclcip3.IRQ115	19	RW	0
qc.mclcip3.IRQ116	20	RW	0
qc.mclcip3.IRQ117	21	RW	0
qc.mclcip3.IRQ118	22	RW	0
qc.mclcip3.IRQ119	23	RW	0
qc.mclcip3.IRQ120	24	RW	0
qc.mclcip3.IRQ121	25	RW	0
qc.mclcip3.IRQ122	26	RW	0
qc.mclcip3.IRQ123	27	RW	0
qc.mclcip3.IRQ124	28	RW	0
qc.mclcip3.IRQ125	29	RW	0
qc.mclcip3.IRQ126	30	RW	0
qc.mclcip3.IRQ127	31	RW	0

5.45.4. Fields

IRQ96

Location

0

Description

IRQ96 pending

Type

RW

Reset value

0

IRQ97

Location

1

Description
<i>IRQ97 pending</i>
Type
<i>RW</i>
Reset value
<i>0</i>

IRQ98

Location
<i>2</i>
Description
<i>IRQ98 pending</i>
Type
<i>RW</i>
Reset value
<i>0</i>

IRQ99

Location
<i>3</i>
Description
<i>IRQ99 pending</i>
Type
<i>RW</i>
Reset value
<i>0</i>

IRQ100

Location
<i>4</i>
Description
<i>IRQ100 pending</i>

Type*RW***Reset value**

0

IRQ101**Location**

5

Description*IRQ101 pending***Type***RW***Reset value**

0

IRQ102**Location**

6

Description*IRQ102 pending***Type***RW***Reset value**

0

IRQ103**Location**

7

Description*IRQ103 pending***Type***RW*

Reset value

0

IRQ104**Location**

8

Description*IRQ104 pending***Type***RW***Reset value**

0

IRQ105**Location**

9

Description*IRQ105 pending***Type***RW***Reset value**

0

IRQ106**Location**

10

Description*IRQ106 pending***Type***RW***Reset value**

0

IRQ107**Location***11***Description***IRQ107 pending***Type***RW***Reset value***0***IRQ108****Location***12***Description***IRQ108 pending***Type***RW***Reset value***0***IRQ109****Location***13***Description***IRQ109 pending***Type***RW***Reset value***0*

IRQ110**Location***14***Description***IRQ110 pending***Type***RW***Reset value***0***IRQ111****Location***15***Description***IRQ111 pending***Type***RW***Reset value***0***IRQ112****Location***16***Description***IRQ112 pending***Type***RW***Reset value***0*

IRQ113**Location***17***Description***IRQ113 pending***Type***RW***Reset value***0***IRQ114****Location***18***Description***IRQ114 pending***Type***RW***Reset value***0***IRQ115****Location***19***Description***IRQ115 pending***Type***RW***Reset value***0*

IRQ116**Location**

20

Description*IRQ116 pending***Type***RW***Reset value**

0

IRQ117**Location**

21

Description*IRQ117 pending***Type***RW***Reset value**

0

IRQ118**Location**

22

Description*IRQ118 pending***Type***RW***Reset value**

0

IRQ119**Location**

23

Description*IRQ119 pending***Type***RW***Reset value**

0

IRQ120**Location**

24

Description*IRQ120 pending***Type***RW***Reset value**

0

IRQ121**Location**

25

Description*IRQ121 pending***Type***RW***Reset value**

0

IRQ122**Location**

26

Description*IRQ122 pending***Type***RW***Reset value**

0

IRQ123**Location**

27

Description*IRQ123 pending***Type***RW***Reset value**

0

IRQ124**Location**

28

Description*IRQ124 pending***Type***RW***Reset value**

0

IRQ125

Location*29***Description***IRQ125 pending***Type***RW***Reset value***0*

IRQ126

Location*30***Description***IRQ126 pending***Type***RW***Reset value***0*

IRQ127

Location*31***Description***IRQ127 pending***Type***RW***Reset value***0*

5.46. qc.mclicip4

IRQ Pending 4

Pending bits for IRQs 128-159

5.46.1. Attributes

Defining Extension	- anyOf: - Xqci, version >= Xqci@0.1.0 - Xqciint, version >= Xqciint@0.1.0
CSR Address	0x7f4
Length	32-bit
Privilege Mode	M

5.46.2. Format

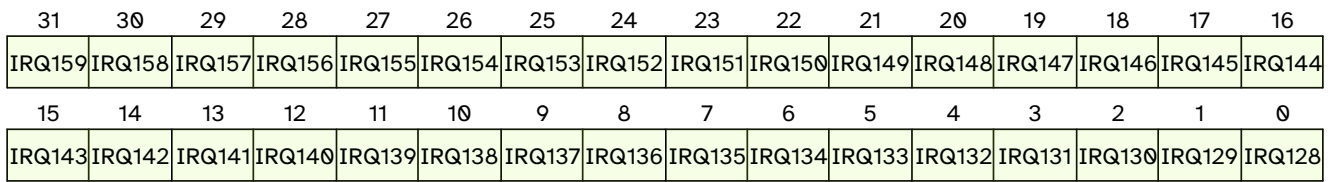


Figure 46. qc.mclicip4 format

5.46.3. Field Summary

Name	Location	Type	Reset Value
qc.mclicip4.IRQ128	0	RW	0
qc.mclicip4.IRQ129	1	RW	0
qc.mclicip4.IRQ130	2	RW	0
qc.mclicip4.IRQ131	3	RW	0
qc.mclicip4.IRQ132	4	RW	0
qc.mclicip4.IRQ133	5	RW	0
qc.mclicip4.IRQ134	6	RW	0
qc.mclicip4.IRQ135	7	RW	0
qc.mclicip4.IRQ136	8	RW	0
qc.mclicip4.IRQ137	9	RW	0
qc.mclicip4.IRQ138	10	RW	0
qc.mclicip4.IRQ139	11	RW	0
qc.mclicip4.IRQ140	12	RW	0
qc.mclicip4.IRQ141	13	RW	0
qc.mclicip4.IRQ142	14	RW	0

Name	Location	Type	Reset Value
qc.mclicip4.IRQ143	15	RW	0
qc.mclicip4.IRQ144	16	RW	0
qc.mclicip4.IRQ145	17	RW	0
qc.mclicip4.IRQ146	18	RW	0
qc.mclicip4.IRQ147	19	RW	0
qc.mclicip4.IRQ148	20	RW	0
qc.mclicip4.IRQ149	21	RW	0
qc.mclicip4.IRQ150	22	RW	0
qc.mclicip4.IRQ151	23	RW	0
qc.mclicip4.IRQ152	24	RW	0
qc.mclicip4.IRQ153	25	RW	0
qc.mclicip4.IRQ154	26	RW	0
qc.mclicip4.IRQ155	27	RW	0
qc.mclicip4.IRQ156	28	RW	0
qc.mclicip4.IRQ157	29	RW	0
qc.mclicip4.IRQ158	30	RW	0
qc.mclicip4.IRQ159	31	RW	0

5.46.4. Fields

IRQ128

Location

0

Description

IRQ128 pending

Type

RW

Reset value

0

IRQ129

Location

1

Description*IRQ129 pending***Type***RW***Reset value**

0

IRQ130**Location**

2

Description*IRQ130 pending***Type***RW***Reset value**

0

IRQ131**Location**

3

Description*IRQ131 pending***Type***RW***Reset value**

0

IRQ132**Location**

4

Description*IRQ132 pending*

Type*RW***Reset value**

0

IRQ133**Location**

5

Description*IRQ133 pending***Type***RW***Reset value**

0

IRQ134**Location**

6

Description*IRQ134 pending***Type***RW***Reset value**

0

IRQ135**Location**

7

Description*IRQ135 pending***Type***RW*

Reset value

0

IRQ136**Location**

8

Description*IRQ136 pending***Type***RW***Reset value**

0

IRQ137**Location**

9

Description*IRQ137 pending***Type***RW***Reset value**

0

IRQ138**Location**

10

Description*IRQ138 pending***Type***RW***Reset value**

0

IRQ139**Location***11***Description***IRQ139 pending***Type***RW***Reset value***0***IRQ140****Location***12***Description***IRQ140 pending***Type***RW***Reset value***0***IRQ141****Location***13***Description***IRQ141 pending***Type***RW***Reset value***0*

IRQ142**Location***14***Description***IRQ142 pending***Type***RW***Reset value***0***IRQ143****Location***15***Description***IRQ143 pending***Type***RW***Reset value***0***IRQ144****Location***16***Description***IRQ144 pending***Type***RW***Reset value***0*

IRQ145**Location***17***Description***IRQ145 pending***Type***RW***Reset value***0***IRQ146****Location***18***Description***IRQ146 pending***Type***RW***Reset value***0***IRQ147****Location***19***Description***IRQ147 pending***Type***RW***Reset value***0*

IRQ148**Location**

20

Description*IRQ148 pending***Type***RW***Reset value**

0

IRQ149**Location**

21

Description*IRQ149 pending***Type***RW***Reset value**

0

IRQ150**Location**

22

Description*IRQ150 pending***Type***RW***Reset value**

0

IRQ151**Location***23***Description***IRQ151 pending***Type***RW***Reset value***0***IRQ152****Location***24***Description***IRQ152 pending***Type***RW***Reset value***0***IRQ153****Location***25***Description***IRQ153 pending***Type***RW***Reset value***0*

IRQ154**Location**

26

Description*IRQ154 pending***Type***RW***Reset value**

0

IRQ155**Location**

27

Description*IRQ155 pending***Type***RW***Reset value**

0

IRQ156**Location**

28

Description*IRQ156 pending***Type***RW***Reset value**

0

IRQ157**Location***29***Description***IRQ157 pending***Type***RW***Reset value***0***IRQ158****Location***30***Description***IRQ158 pending***Type***RW***Reset value***0***IRQ159****Location***31***Description***IRQ159 pending***Type***RW***Reset value***0*

5.47. qc.mclicip5

IRQ Pending 5

Pending bits for IRQs 160-191

5.47.1. Attributes

Defining Extension	- anyOf: - Xqci, version >= Xqci@0.1.0 - Xqciint, version >= Xqciint@0.1.0
CSR Address	0x7f5
Length	32-bit
Privilege Mode	M

5.47.2. Format

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
IRQ191	IRQ190	IRQ189	IRQ188	IRQ187	IRQ186	IRQ185	IRQ184	IRQ183	IRQ182	IRQ181	IRQ180	IRQ179	IRQ178	IRQ177	IRQ176
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IRQ175	IRQ174	IRQ173	IRQ172	IRQ171	IRQ170	IRQ169	IRQ168	IRQ167	IRQ166	IRQ165	IRQ164	IRQ163	IRQ162	IRQ161	IRQ160

Figure 47. qc.mclicip5 format

5.47.3. Field Summary

Name	Location	Type	Reset Value
qc.mclicip5.IRQ160	0	RW	0
qc.mclicip5.IRQ161	1	RW	0
qc.mclicip5.IRQ162	2	RW	0
qc.mclicip5.IRQ163	3	RW	0
qc.mclicip5.IRQ164	4	RW	0
qc.mclicip5.IRQ165	5	RW	0
qc.mclicip5.IRQ166	6	RW	0
qc.mclicip5.IRQ167	7	RW	0
qc.mclicip5.IRQ168	8	RW	0
qc.mclicip5.IRQ169	9	RW	0
qc.mclicip5.IRQ170	10	RW	0
qc.mclicip5.IRQ171	11	RW	0
qc.mclicip5.IRQ172	12	RW	0
qc.mclicip5.IRQ173	13	RW	0
qc.mclicip5.IRQ174	14	RW	0

Name	Location	Type	Reset Value
qc.mclcip5.IRQ175	15	RW	0
qc.mclcip5.IRQ176	16	RW	0
qc.mclcip5.IRQ177	17	RW	0
qc.mclcip5.IRQ178	18	RW	0
qc.mclcip5.IRQ179	19	RW	0
qc.mclcip5.IRQ180	20	RW	0
qc.mclcip5.IRQ181	21	RW	0
qc.mclcip5.IRQ182	22	RW	0
qc.mclcip5.IRQ183	23	RW	0
qc.mclcip5.IRQ184	24	RW	0
qc.mclcip5.IRQ185	25	RW	0
qc.mclcip5.IRQ186	26	RW	0
qc.mclcip5.IRQ187	27	RW	0
qc.mclcip5.IRQ188	28	RW	0
qc.mclcip5.IRQ189	29	RW	0
qc.mclcip5.IRQ190	30	RW	0
qc.mclcip5.IRQ191	31	RW	0

5.47.4. Fields

IRQ160

Location

0

Description

IRQ160 pending

Type

RW

Reset value

0

IRQ161

Location

1

Description*IRQ161 pending***Type***RW***Reset value**

0

IRQ162**Location**

2

Description*IRQ162 pending***Type***RW***Reset value**

0

IRQ163**Location**

3

Description*IRQ163 pending***Type***RW***Reset value**

0

IRQ164**Location**

4

Description*IRQ164 pending*

Type*RW***Reset value***0***IRQ165****Location***5***Description***IRQ165 pending***Type***RW***Reset value***0***IRQ166****Location***6***Description***IRQ166 pending***Type***RW***Reset value***0***IRQ167****Location***7***Description***IRQ167 pending***Type***RW*

Reset value

0

IRQ168**Location**

8

Description*IRQ168 pending***Type***RW***Reset value**

0

IRQ169**Location**

9

Description*IRQ169 pending***Type***RW***Reset value**

0

IRQ170**Location**

10

Description*IRQ170 pending***Type***RW***Reset value**

0

IRQ171**Location***11***Description***IRQ171 pending***Type***RW***Reset value***0***IRQ172****Location***12***Description***IRQ172 pending***Type***RW***Reset value***0***IRQ173****Location***13***Description***IRQ173 pending***Type***RW***Reset value***0*

IRQ174**Location***14***Description***IRQ174 pending***Type***RW***Reset value***0***IRQ175****Location***15***Description***IRQ175 pending***Type***RW***Reset value***0***IRQ176****Location***16***Description***IRQ176 pending***Type***RW***Reset value***0*

IRQ177**Location***17***Description***IRQ177 pending***Type***RW***Reset value***0***IRQ178****Location***18***Description***IRQ178 pending***Type***RW***Reset value***0***IRQ179****Location***19***Description***IRQ179 pending***Type***RW***Reset value***0*

IRQ180**Location***20***Description***IRQ180 pending***Type***RW***Reset value***0***IRQ181****Location***21***Description***IRQ181 pending***Type***RW***Reset value***0***IRQ182****Location***22***Description***IRQ182 pending***Type***RW***Reset value***0*

IRQ183**Location***23***Description***IRQ183 pending***Type***RW***Reset value***0***IRQ184****Location***24***Description***IRQ184 pending***Type***RW***Reset value***0***IRQ185****Location***25***Description***IRQ185 pending***Type***RW***Reset value***0*

IRQ186**Location**

26

Description*IRQ186 pending***Type***RW***Reset value**

0

IRQ187**Location**

27

Description*IRQ187 pending***Type***RW***Reset value**

0

IRQ188**Location**

28

Description*IRQ188 pending***Type***RW***Reset value**

0

IRQ189**Location***29***Description***IRQ189 pending***Type***RW***Reset value***0***IRQ190****Location***30***Description***IRQ190 pending***Type***RW***Reset value***0***IRQ191****Location***31***Description***IRQ191 pending***Type***RW***Reset value***0*

5.48. qc.mclcip6

IRQ Pending 6

Pending bits for IRQs 192-223

5.48.1. Attributes

Defining Extension	- anyOf: - Xqci, version >= Xqci@0.1.0 - Xqciint, version >= Xqciint@0.1.0
CSR Address	0x7f6
Length	32-bit
Privilege Mode	M

5.48.2. Format

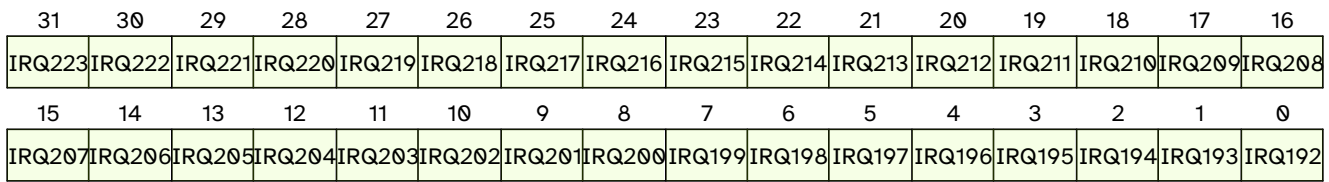


Figure 48. qc.mclcip6 format

5.48.3. Field Summary

Name	Location	Type	Reset Value
qc.mclcip6.IRQ192	0	RW	0
qc.mclcip6.IRQ193	1	RW	0
qc.mclcip6.IRQ194	2	RW	0
qc.mclcip6.IRQ195	3	RW	0
qc.mclcip6.IRQ196	4	RW	0
qc.mclcip6.IRQ197	5	RW	0
qc.mclcip6.IRQ198	6	RW	0
qc.mclcip6.IRQ199	7	RW	0
qc.mclcip6.IRQ200	8	RW	0
qc.mclcip6.IRQ201	9	RW	0
qc.mclcip6.IRQ202	10	RW	0
qc.mclcip6.IRQ203	11	RW	0
qc.mclcip6.IRQ204	12	RW	0
qc.mclcip6.IRQ205	13	RW	0
qc.mclcip6.IRQ206	14	RW	0

Name	Location	Type	Reset Value
qc.mclcip6.IRQ207	15	RW	0
qc.mclcip6.IRQ208	16	RW	0
qc.mclcip6.IRQ209	17	RW	0
qc.mclcip6.IRQ210	18	RW	0
qc.mclcip6.IRQ211	19	RW	0
qc.mclcip6.IRQ212	20	RW	0
qc.mclcip6.IRQ213	21	RW	0
qc.mclcip6.IRQ214	22	RW	0
qc.mclcip6.IRQ215	23	RW	0
qc.mclcip6.IRQ216	24	RW	0
qc.mclcip6.IRQ217	25	RW	0
qc.mclcip6.IRQ218	26	RW	0
qc.mclcip6.IRQ219	27	RW	0
qc.mclcip6.IRQ220	28	RW	0
qc.mclcip6.IRQ221	29	RW	0
qc.mclcip6.IRQ222	30	RW	0
qc.mclcip6.IRQ223	31	RW	0

5.48.4. Fields

IRQ192

Location

0

Description

IRQ192 pending

Type

RW

Reset value

0

IRQ193

Location

1

Description*IRQ193 pending***Type***RW***Reset value**

0

IRQ194**Location**

2

Description*IRQ194 pending***Type***RW***Reset value**

0

IRQ195**Location**

3

Description*IRQ195 pending***Type***RW***Reset value**

0

IRQ196**Location**

4

Description*IRQ196 pending*

Type*RW***Reset value***0***IRQ197****Location***5***Description***IRQ197 pending***Type***RW***Reset value***0***IRQ198****Location***6***Description***IRQ198 pending***Type***RW***Reset value***0***IRQ199****Location***7***Description***IRQ199 pending***Type***RW*

Reset value

0

IRQ200**Location**

8

Description*IRQ200 pending***Type***RW***Reset value**

0

IRQ201**Location**

9

Description*IRQ201 pending***Type***RW***Reset value**

0

IRQ202**Location**

10

Description*IRQ202 pending***Type***RW***Reset value**

0

IRQ203**Location***11***Description***IRQ203 pending***Type***RW***Reset value***0***IRQ204****Location***12***Description***IRQ204 pending***Type***RW***Reset value***0***IRQ205****Location***13***Description***IRQ205 pending***Type***RW***Reset value***0*

IRQ206**Location***14***Description***IRQ206 pending***Type***RW***Reset value***0***IRQ207****Location***15***Description***IRQ207 pending***Type***RW***Reset value***0***IRQ208****Location***16***Description***IRQ208 pending***Type***RW***Reset value***0*

IRQ209**Location***17***Description***IRQ209 pending***Type***RW***Reset value***0***IRQ210****Location***18***Description***IRQ210 pending***Type***RW***Reset value***0***IRQ211****Location***19***Description***IRQ211 pending***Type***RW***Reset value***0*

IRQ212

Location

20

Description*IRQ212 pending***Type***RW***Reset value**

0

IRQ213

Location

21

Description*IRQ213 pending***Type***RW***Reset value**

0

IRQ214

Location

22

Description*IRQ214 pending***Type***RW***Reset value**

0

IRQ215

Location

23

Description*IRQ215 pending***Type***RW***Reset value**

0

IRQ216

Location

24

Description*IRQ216 pending***Type***RW***Reset value**

0

IRQ217

Location

25

Description*IRQ217 pending***Type***RW***Reset value**

0

IRQ218**Location**

26

Description*IRQ218 pending***Type***RW***Reset value**

0

IRQ219**Location**

27

Description*IRQ219 pending***Type***RW***Reset value**

0

IRQ220**Location**

28

Description*IRQ220 pending***Type***RW***Reset value**

0

IRQ221

Location*29***Description***IRQ221 pending***Type***RW***Reset value***0*

IRQ222

Location*30***Description***IRQ222 pending***Type***RW***Reset value***0*

IRQ223

Location*31***Description***IRQ223 pending***Type***RW***Reset value***0*

5.49. qc.mclicip7

IRQ Pending 7

Pending bits for IRQs 224-255

5.49.1. Attributes

Defining Extension	- anyOf: - Xqci, version >= Xqci@0.1.0 - Xqciint, version >= Xqciint@0.1.0
CSR Address	0x7f7
Length	32-bit
Privilege Mode	M

5.49.2. Format

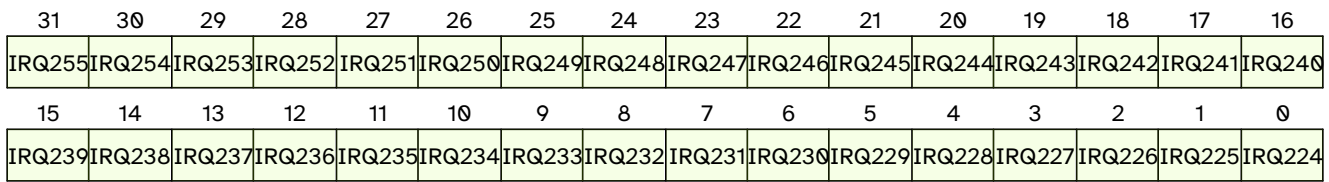


Figure 49. qc.mclicip7 format

5.49.3. Field Summary

Name	Location	Type	Reset Value
qc.mclicip7.IRQ224	0	RW	0
qc.mclicip7.IRQ225	1	RW	0
qc.mclicip7.IRQ226	2	RW	0
qc.mclicip7.IRQ227	3	RW	0
qc.mclicip7.IRQ228	4	RW	0
qc.mclicip7.IRQ229	5	RW	0
qc.mclicip7.IRQ230	6	RW	0
qc.mclicip7.IRQ231	7	RW	0
qc.mclicip7.IRQ232	8	RW	0
qc.mclicip7.IRQ233	9	RW	0
qc.mclicip7.IRQ234	10	RW	0
qc.mclicip7.IRQ235	11	RW	0
qc.mclicip7.IRQ236	12	RW	0
qc.mclicip7.IRQ237	13	RW	0
qc.mclicip7.IRQ238	14	RW	0

Name	Location	Type	Reset Value
qc.mclicip7.IRQ239	15	RW	0
qc.mclicip7.IRQ240	16	RW	0
qc.mclicip7.IRQ241	17	RW	0
qc.mclicip7.IRQ242	18	RW	0
qc.mclicip7.IRQ243	19	RW	0
qc.mclicip7.IRQ244	20	RW	0
qc.mclicip7.IRQ245	21	RW	0
qc.mclicip7.IRQ246	22	RW	0
qc.mclicip7.IRQ247	23	RW	0
qc.mclicip7.IRQ248	24	RW	0
qc.mclicip7.IRQ249	25	RW	0
qc.mclicip7.IRQ250	26	RW	0
qc.mclicip7.IRQ251	27	RW	0
qc.mclicip7.IRQ252	28	RW	0
qc.mclicip7.IRQ253	29	RW	0
qc.mclicip7.IRQ254	30	RW	0
qc.mclicip7.IRQ255	31	RW	0

5.49.4. Fields

IRQ224

Location

0

Description

IRQ224 pending

Type

RW

Reset value

0

IRQ225

Location

1

Description
<i>IRQ225 pending</i>
Type
<i>RW</i>
Reset value
<i>0</i>

IRQ226

Location
<i>2</i>
Description
<i>IRQ226 pending</i>
Type
<i>RW</i>
Reset value
<i>0</i>

IRQ227

Location
<i>3</i>
Description
<i>IRQ227 pending</i>
Type
<i>RW</i>
Reset value
<i>0</i>

IRQ228

Location
<i>4</i>
Description
<i>IRQ228 pending</i>

Type*RW***Reset value**

0

IRQ229**Location**

5

Description*IRQ229 pending***Type***RW***Reset value**

0

IRQ230**Location**

6

Description*IRQ230 pending***Type***RW***Reset value**

0

IRQ231**Location**

7

Description*IRQ231 pending***Type***RW*

Reset value

0

IRQ232**Location**

8

Description*IRQ232 pending***Type***RW***Reset value**

0

IRQ233**Location**

9

Description*IRQ233 pending***Type***RW***Reset value**

0

IRQ234**Location**

10

Description*IRQ234 pending***Type***RW***Reset value**

0

IRQ235

Location*11***Description***IRQ235 pending***Type***RW***Reset value***0*

IRQ236

Location*12***Description***IRQ236 pending***Type***RW***Reset value***0*

IRQ237

Location*13***Description***IRQ237 pending***Type***RW***Reset value***0*

IRQ238**Location***14***Description***IRQ238 pending***Type***RW***Reset value***0***IRQ239****Location***15***Description***IRQ239 pending***Type***RW***Reset value***0***IRQ240****Location***16***Description***IRQ240 pending***Type***RW***Reset value***0*

IRQ241

Location*17***Description***IRQ241 pending***Type***RW***Reset value***0*

IRQ242

Location*18***Description***IRQ242 pending***Type***RW***Reset value***0*

IRQ243

Location*19***Description***IRQ243 pending***Type***RW***Reset value***0*

IRQ244

Location

20

Description*IRQ244 pending***Type***RW***Reset value**

0

IRQ245

Location

21

Description*IRQ245 pending***Type***RW***Reset value**

0

IRQ246

Location

22

Description*IRQ246 pending***Type***RW***Reset value**

0

IRQ247

Location

23

Description*IRQ247 pending***Type***RW***Reset value**

0

IRQ248

Location

24

Description*IRQ248 pending***Type***RW***Reset value**

0

IRQ249

Location

25

Description*IRQ249 pending***Type***RW***Reset value**

0

IRQ250**Location**

26

Description*IRQ250 pending***Type***RW***Reset value**

0

IRQ251**Location**

27

Description*IRQ251 pending***Type***RW***Reset value**

0

IRQ252**Location**

28

Description*IRQ252 pending***Type***RW***Reset value**

0

IRQ253

Location*29***Description***IRQ253 pending***Type***RW***Reset value***0*

IRQ254

Location*30***Description***IRQ254 pending***Type***RW***Reset value***0*

IRQ255

Location*31***Description***IRQ255 pending***Type***RW***Reset value***0*

5.50. qc.mmcr

Machine Mode Control Register

Machine Mode Control Register.

5.50.1. Attributes

Defining Extension	- anyOf: - Xqci, version >=0.7 - Xqciint, version >=0.4
CSR Address	0x7c0
Length	32-bit
Privilege Mode	M

5.50.2. Format

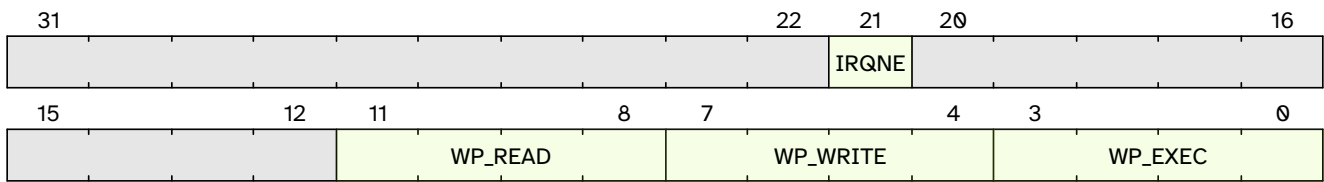


Figure 50. qc.mmcr format

5.50.3. Field Summary

Name	Location	Type	Reset Value
qc.mmcr.WP_EXEC	3:0	RW	0
qc.mmcr.WP_WRITE	7:4	RW	0
qc.mmcr.WP_READ	11:8	RW	0
qc.mmcr.IRQNE	21	RW	0

5.50.4. Fields

WP_EXEC

Location 3:0
Description Watchpoint execution enable (WP 0-3)
Type RW

Reset value

0

WP_WRITE

Location

7:4

Description

Watchpoint write enable (WP 0-3)

Type

RW

Reset value

0

WP_READ

Location

11:8

Description

Watchpoint read enable (WP 0-3)

Type

RW

Reset value

0

IRQNE

Location

21

Description

IRQ nesting enable (for custom IRQ only)

Type

RW

Reset value

0

5.51. qc.mntvec

Machine Non-Maskable Interrupt Vector Control

Controls where NMI jump.

5.51.1. Attributes

Defining Extension	- anyOf: - Xqci, version >= Xqci@0.1.0 - Xqciint, version >= Xqciint@0.1.0
CSR Address	0x7c3
Length	32-bit
Privilege Mode	M

5.51.2. Format

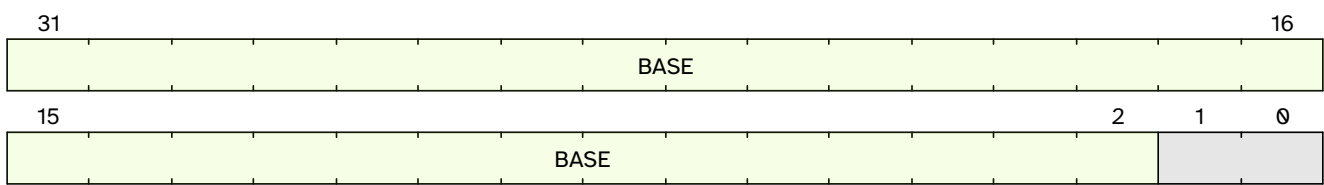


Figure 51. qc.mntvec format

5.51.3. Field Summary

Name	Location	Type	Reset Value
qc.mntvec.BASE	31:2	RW-R	0

5.51.4. Fields

BASE

Location 31:2
Description Bits [MXLEN-1:2] of the NMI vector physical address for any NMI taken in M-mode.
Type RW-R
Reset value 0

5.51.5. Software write

This CSR may store a value that is different from what software attempts to write.

When a software write occurs (e.g., through [csrrw](#)), the following determines the written value:

```
BASE = return csr_value.BASE;
```

5.52. qc.mstkbottomaddr

Machine Stack Bottom Limit

Stack bottom limit register. If the stack pointer ($sp == x2$) lesser then its value - SpOutOfRangeException exception occurs

5.52.1. Attributes

Defining Extension	- anyOf: - Xqci, version ≥ 0.7 - Xqciint, version ≥ 0.4
CSR Address	0x7c5
Length	32-bit
Privilege Mode	M

5.52.2. Format

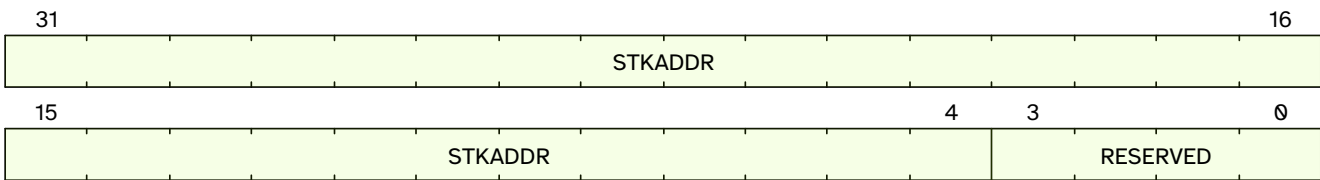


Figure 52. qc.mstkbottomaddr format

5.52.3. Field Summary

Name	Location	Type	Reset Value
qc.mstkbottomaddr.STKADDR	31:4	RW	0
qc.mstkbottomaddr.RESERVED	3:0	RO	0

5.52.4. Fields

STKADDR

Location
31:4

Description
If the stack pointer ($sp == x2$) lesser then $STKADDR * 16$ - SpOutOfRangeException exception occurs

Type
RW

Reset value

0

RESERVED

Location

3:0

Description

Reserved and must be zero

Type

RO

Reset value

0

5.53. qc.mstktopaddr

Machine Stack Top Limit

Stack top limit register. If the stack pointer (sp == x2) greater then its value - SpOutOfRangeException exception occurs

5.53.1. Attributes

Defining Extension	- anyOf: - Xqci, version >=0.7 - Xqciint, version >=0.4
CSR Address	0x7c4
Length	32-bit
Privilege Mode	M

5.53.2. Format

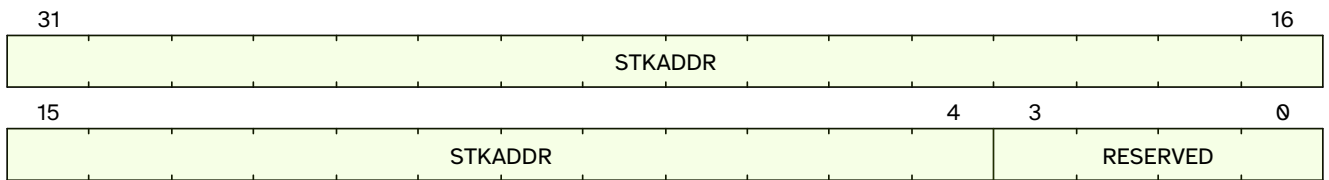


Figure 53. qc.mstktopaddr format

5.53.3. Field Summary

Name	Location	Type	Reset Value
qc.mstktopaddr.STKADDR	31:4	RW	0
qc.mstktopaddr.RESERVED	3:0	RO	0

5.53.4. Fields

STKADDR

Location

31:4

Description

If the stack pointer (sp == x2) greater then STKADDR*16 - SpOutOfRangeException exception occurs

Type

RW

Reset value

0

RESERVED

Location

3:0

Description

Reserved and must be zero

Type

RO

Reset value

0

5.54. qc.mthreadptr

Machine Thread Pointer Register

Thread pointer register for software use in RTOS. Bits are not interpreted by hardware.

5.54.1. Attributes

Defining Extension	- anyOf: - Xqci, version >=0.7 - Xqciint, version >=0.4
CSR Address	0x7c8
Length	32-bit
Privilege Mode	M

5.54.2. Format

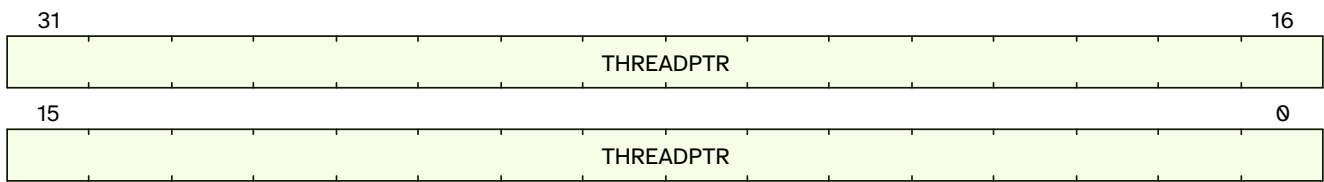


Figure 54. qc.mthreadptr format

5.54.3. Field Summary

Name	Location	Type	Reset Value
qc.mthreadptr.THREADPTR	31:0	RW	0

5.54.4. Fields

THREADPTR

Location 31:0
Description Thread pointer value
Type RW
Reset value 0

5.55. qc.mwpendaddr0

Watchpoint end address for region 0

Watchpoint end address for region 0

5.55.1. Attributes

Defining Extension	- anyOf: - Xqci, version >=0.7 - Xqciint, version >=0.4
CSR Address	0x7d4
Length	32-bit
Privilege Mode	M

5.55.2. Format

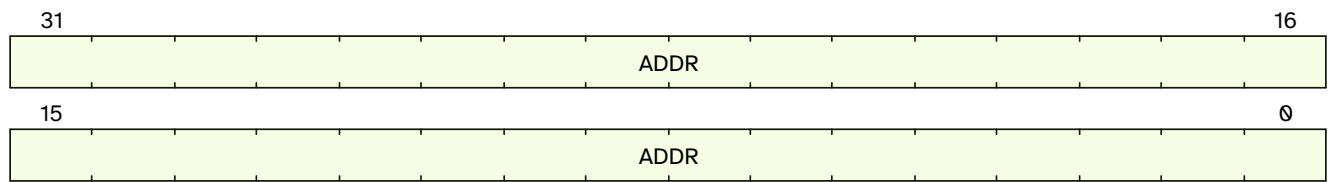


Figure 55. qc.mwpendaddr0 format

5.55.3. Field Summary

Name	Location	Type	Reset Value
qc.mwpendaddr0.ADDR	31:0	RW	0

5.55.4. Fields

ADDR

Location

31:0

Description

Watchpoint end address

Type

RW

Reset value

0

5.56. qc.mwpendaddr1

Watchpoint end address for region 1

Watchpoint end address for region 1

5.56.1. Attributes

Defining Extension	- anyOf: - Xqci, version >=0.7 - Xqciint, version >=0.4
CSR Address	0x7d5
Length	32-bit
Privilege Mode	M

5.56.2. Format

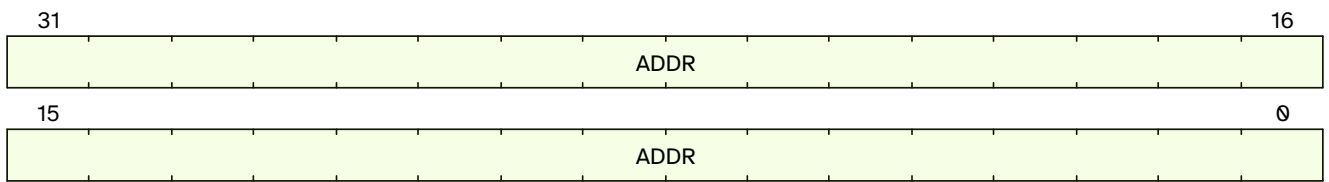


Figure 56. qc.mwpendaddr1 format

5.56.3. Field Summary

Name	Location	Type	Reset Value
qc.mwpendaddr1.ADDR	31:0	RW	0

5.56.4. Fields

ADDR

Location

31:0

Description

Watchpoint end address

Type

RW

Reset value

0

5.57. qc.mwpendaddr2

Watchpoint end address for region 2

Watchpoint end address for region 2

5.57.1. Attributes

Defining Extension	- anyOf: - Xqci, version >=0.7 - Xqciint, version >=0.4
CSR Address	0x7d6
Length	32-bit
Privilege Mode	M

5.57.2. Format

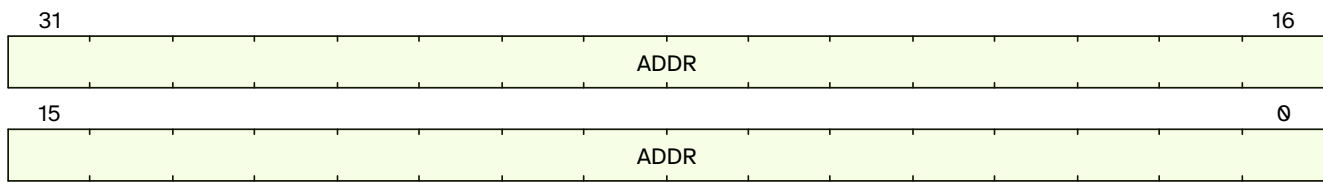


Figure 57. qc.mwpendaddr2 format

5.57.3. Field Summary

Name	Location	Type	Reset Value
qc.mwpendaddr2.ADDR	31:0	RW	0

5.57.4. Fields

ADDR

Location 31:0
Description Watchpoint end address
Type RW
Reset value 0

5.58. qc.mwpendaddr3

Watchpoint end address for region 3

Watchpoint end address for region 3

5.58.1. Attributes

Defining Extension	- anyOf: - Xqci, version >=0.7 - Xqciint, version >=0.4
CSR Address	0x7d7
Length	32-bit
Privilege Mode	M

5.58.2. Format

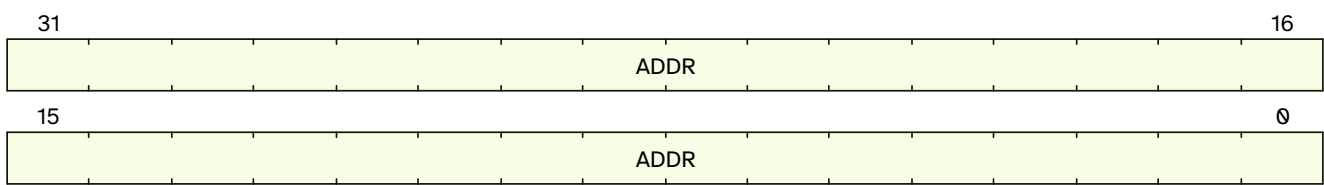


Figure 58. qc.mwpendaddr3 format

5.58.3. Field Summary

Name	Location	Type	Reset Value
qc.mwpendaddr3.ADDR	31:0	RW	0

5.58.4. Fields

ADDR

Location 31:0
Description Watchpoint end address
Type RW
Reset value 0

5.59. qc.mwpstartaddr0

Watchpoint start address for region 0

Watchpoint start address for region 0

5.59.1. Attributes

Defining Extension	- anyOf: - Xqci, version >=0.7 - Xqciint, version >=0.4
CSR Address	0x7d0
Length	32-bit
Privilege Mode	M

5.59.2. Format

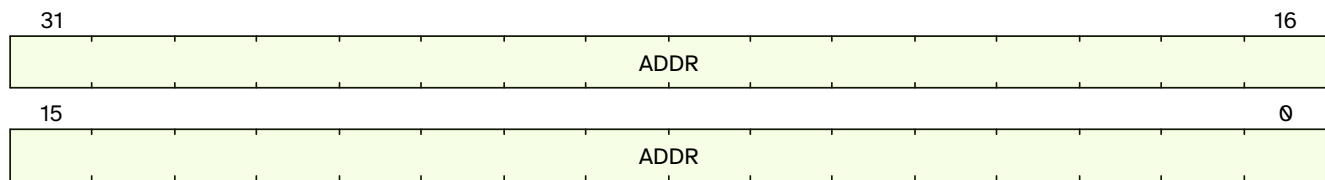


Figure 59. qc.mwpstartaddr0 format

5.59.3. Field Summary

Name	Location	Type	Reset Value
qc.mwpstartaddr0.ADDR	31:0	RW	0

5.59.4. Fields

ADDR

Location 31:0
Description Watchpoint start address
Type RW
Reset value 0

5.60. qc.mwpstartaddr1

Watchpoint start address for region 1

Watchpoint start address for region 1

5.60.1. Attributes

Defining Extension	- anyOf: - Xqci, version >=0.7 - Xqciint, version >=0.4
CSR Address	0x7d1
Length	32-bit
Privilege Mode	M

5.60.2. Format

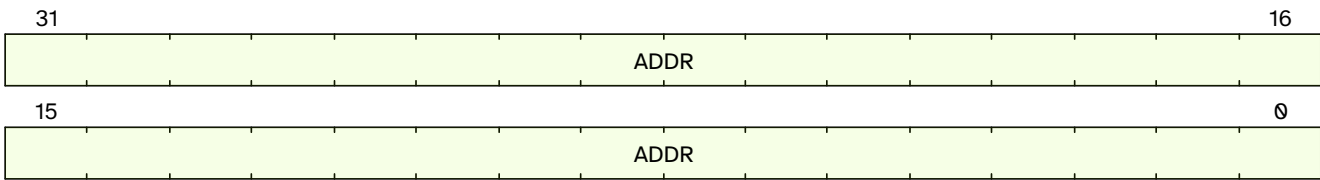


Figure 60. qc.mwpstartaddr1 format

5.60.3. Field Summary

Name	Location	Type	Reset Value
qc.mwpstartaddr1.ADDR	31:0	RW	0

5.60.4. Fields

ADDR

Location 31:0
Description Watchpoint start address
Type RW
Reset value 0

5.61. qc.mwpstartaddr2

Watchpoint start address for region 2

Watchpoint start address for region 2

5.61.1. Attributes

Defining Extension	- anyOf: - Xqci, version ≥ 0.7 - Xqciint, version ≥ 0.4
CSR Address	0x7d2
Length	32-bit
Privilege Mode	M

5.61.2. Format

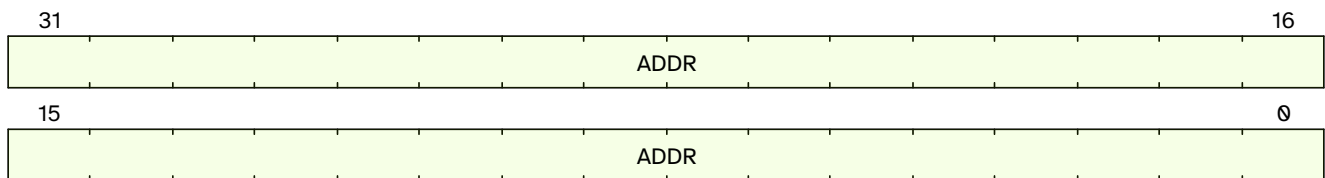


Figure 61. qc.mwpstartaddr2 format

5.61.3. Field Summary

Name	Location	Type	Reset Value
qc.mwpstartaddr2.ADDR	31:0	RW	0

5.61.4. Fields

ADDR

Location

31:0

Description

Watchpoint start address

Type

RW

Reset value

0

5.62. qc.mwpstartaddr3

Watchpoint start address for region 3

Watchpoint start address for region 3

5.62.1. Attributes

Defining Extension	- anyOf: - Xqci, version >=0.7 - Xqciint, version >=0.4
CSR Address	0x7d3
Length	32-bit
Privilege Mode	M

5.62.2. Format



Figure 62. qc.mwpstartaddr3 format

5.62.3. Field Summary

Name	Location	Type	Reset Value
qc.mwpstartaddr3.ADDR	31:0	RW	0

5.62.4. Fields

ADDR

Location 31:0
Description Watchpoint start address
Type RW
Reset value 0

Chapter 6. Instructions (in alphabetical order)

6.1. qc.addsat

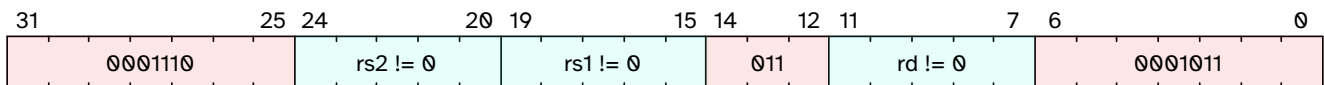
Synopsis

Saturating signed addition

Mnemonic

```
qc.addsat rd, rs1, rs2
```

Encoding



Description

Add signed values `rs1` and `rs2`, saturate the signed result, and write to `rd`. Instruction encoded in R instruction format

Decode Variables

```
Bits<5> rs1 = $encoding[19:15];
Bits<5> rs2 = $encoding[24:20];
Bits<5> rd = $encoding[11:7];
```

Operation

```
XReg sum = X[rs1] + X[rs2];
XReg most_negative_number = 1 << (xlen() - 1);
XReg most_positive_number = (1 << (xlen() - 1)) - 1;
if (X[rs1][xlen() - 1] == X[rs2][xlen() - 1]) {
    if (sum[xlen() - 1] != X[rs1][xlen() - 1]) {
        if ($signed(X[rs1]) < 0) {
            sum = most_negative_number;
        } else {
            sum = most_positive_number;
        }
    }
}
X[rd] = sum;
```

Included in

Extension	Version
Xqci	~> 0.1.0
Xqcia	~> 0.1.0

6.2. qc.addusat

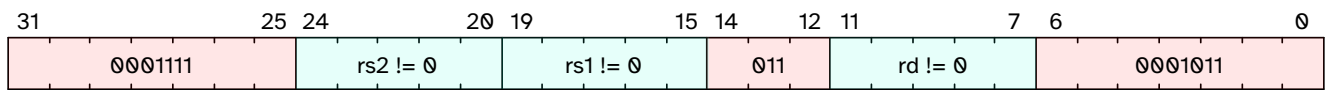
Synopsis

Saturating unsigned addition

Mnemonic

```
qc.addusat rd, rs1, rs2
```

Encoding



Description

Add unsigned values `rs1` and `rs2`, saturate the unsigned result, and write to `rd`. Instruction encoded in R instruction format

Decode Variables

```
Bits<5> rs1 = $encoding[19:15];
Bits<5> rs2 = $encoding[24:20];
Bits<5> rd = $encoding[11:7];
```

Operation

```
XReg sum = X[rs1] + X[rs2];
if ((X[rs1][xlen() - 1] == 1) || (X[rs2][xlen() - 1] == 1)) {
    if (sum[xlen() - 1] == 0) {
        sum = ~{MXLEN{1'b0}};
    }
}
X[rd] = sum;
```

Included in

Extension	Version
Xqci	~> 0.1.0
Xqcia	~> 0.1.0

6.3. qc.beqi

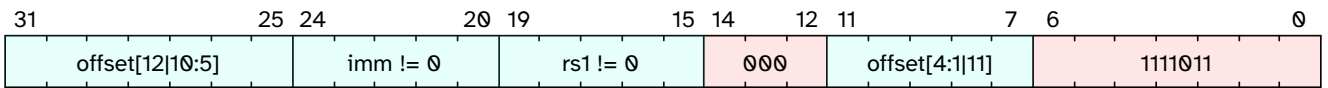
Synopsis

Branch on equal (Immediate)

Mnemonic

```
qc.beqi  rs1, imm, offset
```

Encoding



Description

Branches to PC + offset if the value in rs1 is equal to the signed immediate Instruction encoded in BI instruction format

Decode Variables

```
signed Bits<13> offset = sext({$encoding[31], $encoding[7], $encoding[30:25],  
$encoding[11:8], 1'd0});  
Bits<5> imm = $encoding[24:20];  
Bits<5> rs1 = $encoding[19:15];
```

Operation

```
if ($signed(X[rs1]) == $signed(imm)) {  
    jump_halfword($pc + $signed(offset));  
}
```

Included in

Extension	Version
Xqci	~> 0.1.0
Xqcibi	~> 0.1.0

6.4. qc.bgei

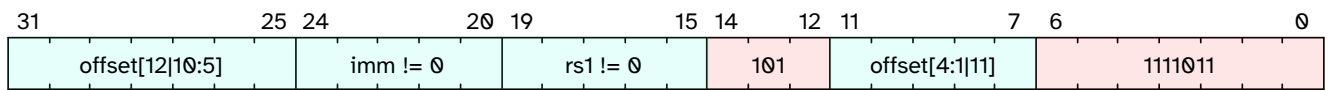
Synopsis

Branch on greater than or equal (immediate)

Mnemonic

```
qc.bgei  rs1, imm, offset
```

Encoding



Description

Branches to PC + offset if the value in rs1 is greater than or equal to the immediate. Instruction encoded in BI instruction format

Decode Variables

```
signed Bits<13> offset = sext({$encoding[31], $encoding[7], $encoding[30:25],  
$encoding[11:8], 1'd0});  
Bits<5> imm = $encoding[24:20];  
Bits<5> rs1 = $encoding[19:15];
```

Operation

```
if ($signed(X[rs1]) >= $signed(imm)) {  
    jump_halfword($pc + $signed(offset));  
}
```

Included in

Extension	Version
Xqci	~> 0.1.0
Xqcibi	~> 0.1.0

6.5. qc.bgeui

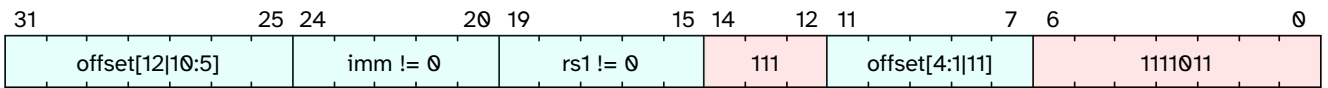
Synopsis

Branch on greater than or equal unsigned (immediate)

Mnemonic

```
qc.bgeui  rs1, imm, offset
```

Encoding



Description

Branches to PC + offset if the unsigned value in rs1 is greater than or equal to the unsigned immediate. Instruction encoded in BI instruction format

Decode Variables

```
signed Bits<13> offset = sext({$encoding[31], $encoding[7], $encoding[30:25],  
$encoding[11:8], 1'd0});  
Bits<5> imm = $encoding[24:20];  
Bits<5> rs1 = $encoding[19:15];
```

Operation

```
if (X[rs1] >= imm) {  
    jump_halfword($pc + $signed(offset));  
}
```

Included in

Extension	Version
Xqci	~> 0.1.0
Xqcibi	~> 0.1.0

6.6. qc.blti

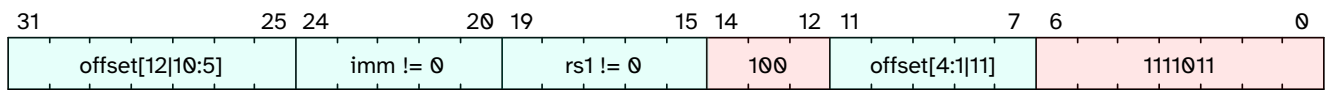
Synopsis

Branch on less than (immediate)

Mnemonic

```
qc.blti  rs1, imm, offset
```

Encoding



Description

Branches to PC + offset if the value in rs1 is less than the immediate. Instruction encoded in BI instruction format

Decode Variables

```
signed Bits<13> offset = sext({$encoding[31], $encoding[7], $encoding[30:25],  
$encoding[11:8], 1'd0});  
Bits<5> imm = $encoding[24:20];  
Bits<5> rs1 = $encoding[19:15];
```

Operation

```
if ($signed(X[rs1]) < $signed(imm)) {  
    jump_halfword($pc + $signed(offset));  
}
```

Included in

Extension	Version
Xqci	~> 0.1.0
Xqcibi	~> 0.1.0

6.7. qc.bltoi

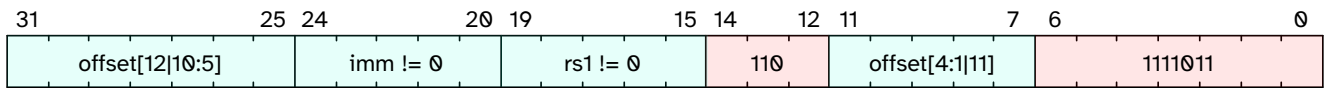
Synopsis

Branch on less than unsigned (immediate)

Mnemonic

```
qc.bltoi  rs1, imm, offset
```

Encoding



Description

Branches to PC + offset if the unsigned value in rs1 is less than the unsigned immediate. Instruction encoded in BI instruction format

Decode Variables

```
signed Bits<13> offset = sext({$encoding[31], $encoding[7], $encoding[30:25],  
$encoding[11:8], 1'd0});  
Bits<5> imm = $encoding[24:20];  
Bits<5> rs1 = $encoding[19:15];
```

Operation

```
if (X[rs1] < imm) {  
    jump_halfword($pc + $signed(offset));  
}
```

Included in

Extension	Version
Xqci	~> 0.1.0
Xqcibi	~> 0.1.0

6.8. qc.bnei

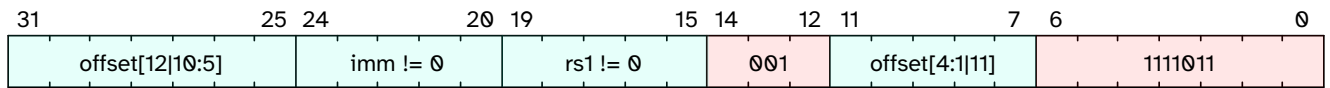
Synopsis

Branch on not equal (immediate)

Mnemonic

```
qc.bnei  rs1, imm, offset
```

Encoding



Description

Branches to PC + offset if the value in rs1 is not equal to the signed immediate. Instruction encoded in BI instruction format.

Decode Variables

```
signed Bits<13> offset = sext({$encoding[31], $encoding[7], $encoding[30:25],  
$encoding[11:8], 1'd0});  
Bits<5> imm = $encoding[24:20];  
Bits<5> rs1 = $encoding[19:15];
```

Operation

```
if ($signed(X[rs1]) != $signed(imm)) {  
    jump_halfword($pc + $signed(offset));  
}
```

Included in

Extension	Version
Xqci	~> 0.1.0
Xqcibi	~> 0.1.0

6.9. qc.brev32

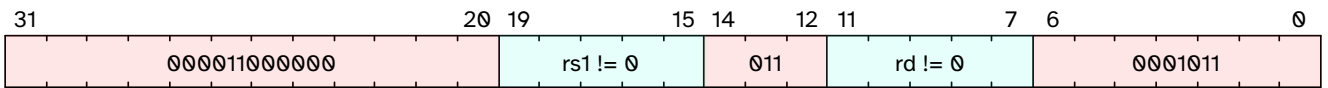
Synopsis

Reverse bit order

Mnemonic

```
qc.brev32 rd, rs1
```

Encoding



Description

Reverses the bit order of `rs1` and writes the result to `rd`. Instruction encoded in I instruction format

Decode Variables

```
Bits<5> rs1 = $encoding[19:15];
Bits<5> rd = $encoding[11:7];
```

Operation

```
XReg orig_val = X[rs1];
XReg tmp;
for (U32 i = 0; i < xlen(); i++) {
    tmp[xlen() - 1 - i] = orig_val[i];
}
X[rd] = tmp;
```

Included in

Extension	Version
Xqci	~> 0.1.0
Xqcibm	~> 0.1.0

6.10. qc.c.bexti

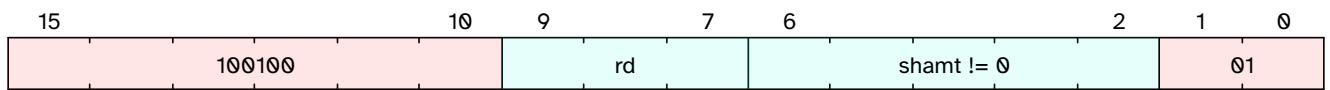
Synopsis

Single-Bit extract (Immediate)

Mnemonic

```
qc.c.bexti rd, shamt
```

Encoding



Description

This instruction returns a single bit extracted from `rd`. The index is read from the lower $\log_2(XLEN)$ bits of `shamt`. Instruction encoded in CB instruction format.

Decode Variables

```
Bits<5> shamt = $encoding[6:2];
Bits<3> rd = $encoding[9:7];
```

Operation

```
XReg index = shamt & (xlen() - 1);
XReg reg = creg2reg(rd);
X[reg] = (X[reg] >> index) & 1;
```

Included in

Extension	Version
Xqci	~> 0.1.0
Xqcibm	~> 0.1.0

6.11. qc.c.bseti

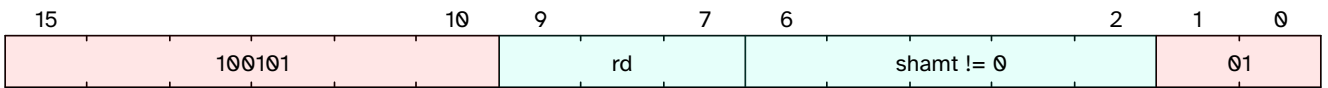
Synopsis

Single-Bit set (Immediate)

Mnemonic

```
qc.c.bseti rd, shamt
```

Encoding



Description

This instruction returns `rd` with a single bit set at the index specified in `shamt`. The index is read from the lower $\log_2(\text{XLEN})$ bits of `shamt`. Instruction encoded in CB instruction format.

Decode Variables

```
Bits<5> shamt = $encoding[6:2];
Bits<3> rd = $encoding[9:7];
```

Operation

```
XReg index = shamt & (xlen() - 1);
XReg reg = creg2reg(rd);
X[reg] = X[reg] | (1 << index);
```

Included in

Extension	Version
Xqci	~> 0.1.0
Xqcibm	~> 0.1.0

6.12. qc.c.clrint

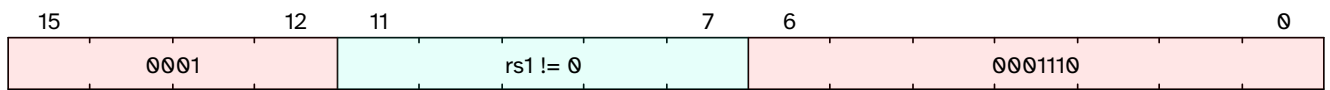
Synopsis

Clear interrupt (Register)

Mnemonic

```
qc.c.clrint  rs1
```

Encoding



Description

Clear interrupt, interrupt number is in rs1. Instruction encoded in CI instruction format.

Decode Variables

```
Bits<5> rs1 = $encoding[11:7];
```

Operation

```
Bits<12> MCLICIP0_ADDR = CSR[qc.mclicip0].address();
XReg idx = rs1 / 32;
XReg bit = rs1 % 32;
XReg pre_csr = CSR[MCLICIP0_ADDR + idx].sw_read();
CSR[MCLICIP0_ADDR + idx].sw_write(pre_csr & ~(1 << bit));
```

Included in

Extension	Version
Xqci	~> 0.1.0
Xqciint	~> 0.1.0

6.13. qc.c.delay

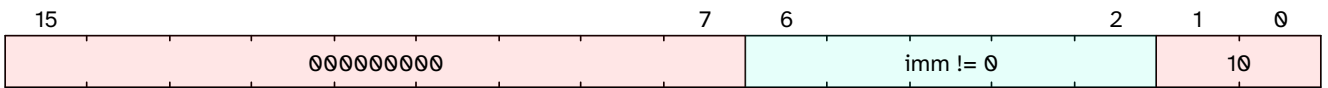
Synopsis

Delay execution for immediate amount of cycles

Mnemonic

```
qc.c.delay
```

Encoding



Description

Delay execution for amount of cycles provided as immediate argument Instruction encoded in CI instruction format.

Decode Variables

```
Bits<5> imm = $encoding[6:2];
```

Operation

```
delay(imm);
```

Included in

Extension	Version
Xqci	~> 0.1.0
Xqcisync	~> 0.1.0

6.14. qc.c.di

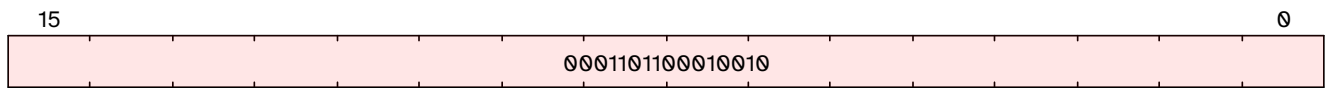
Synopsis

Disable interrupts

Mnemonic

qc.c.di

Encoding



Description

Globally disable interrupts. Equivalent to "csrrci zero, [mstatus](#), 8". Instruction encoded in CI instruction format.

Decode Variables

qc.c.di has no decode variables.

Operation

```
CSR[mstatus].MIE = 0;
XReg pre_qc_mcause = CSR[qc.mcause].sw_read();
CSR[qc.mcause].sw_write(pre_qc_mcause & ~(1 << 26));
```

Included in

Extension	Version
Xqci	~> 0.1.0
Xqciint	~> 0.1.0

6.15. qc.c.dir

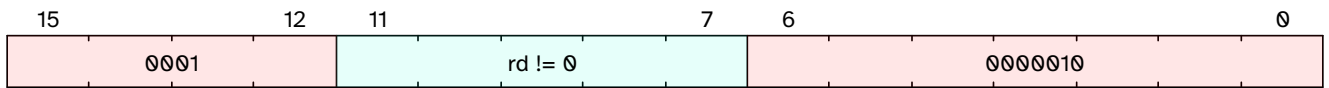
Synopsis

Disable interrupts (Register)

Mnemonic

```
qc.c.dir rd
```

Encoding



Description

Globally disable interrupts, write previous value of `mstatus` to `rd`. Equivalent to "csrrci `rd`, `mstatus`, 8". Instruction encoded in CI instruction format.

Decode Variables

```
Bits<5> rd = $encoding[11:7];
```

Operation

```
XReg pre_mstatus = CSR[mstatus].sw_read();
CSR[mstatus].MIE = 0;
XReg pre_qc_mcause = CSR[qc.mcause].sw_read();
CSR[qc.mcause].sw_write(pre_qc_mcause & ~(1 << 26));
X[rd] = pre_mstatus;
```

Included in

Extension	Version
Xqci	~> 0.1.0
Xqciint	~> 0.1.0

6.16. qc.c.ei

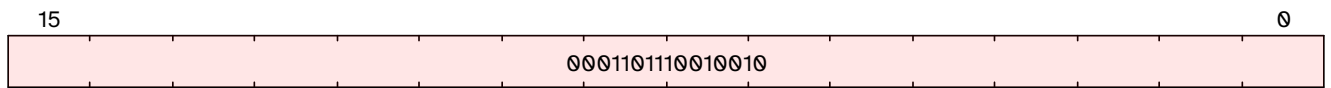
Synopsis

Enable interrupts

Mnemonic

qc.c.ei

Encoding



Description

Globally enable interrupts. Equivalent to "csrrsi zero, mstatus, 8". Instruction encoded in CI instruction format.

Decode Variables

qc.c.ei has no decode variables.

Operation

```
CSR[mstatus].MIE = 1;
XReg pre_qc_mcause = CSR[qc.mcause].sw_read();
CSR[qc.mcause].sw_write(pre_qc_mcause | (1 << 26));
```

Included in

Extension	Version
Xqci	~> 0.1.0
Xqciint	~> 0.1.0

6.17. qc.c.eir

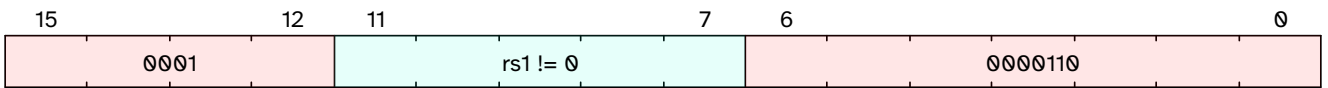
Synopsis

Restore interrupts (Register)

Mnemonic

```
qc.c.eir  rs1
```

Encoding



Description

Globally restore interrupts, write `rs1` to `mstatus`. Equivalent to "csrrs zero, `mstatus`, ``rs1``". Instruction encoded in CI instruction format.

Decode Variables

```
Bits<5> rs1 = $encoding[11:7];
```

Operation

```
Bits<1> mie_val = (X[rs1] >> 3) & 1;
CSR[mstatus].MIE = mie_val;
XReg pre_qc_mcause = CSR[qc.mcause].sw_read();
XReg pre_qc_mcause_masked = pre_qc_mcause & ~(1 << 26);
CSR[qc.mcause].sw_write(pre_qc_mcause | (mie_val << 26));
```

Included in

Extension	Version
Xqci	~> 0.1.0
Xqciint	~> 0.1.0

6.18. qc.c.extu

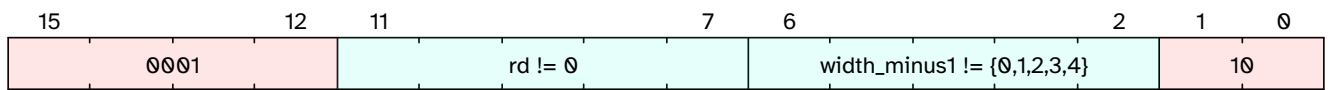
Synopsis

Extract bits unsigned

Mnemonic

```
qc.c.extu rd, width
```

Encoding



Description

Extract a subset of bits from `rd` starting from LSB. The width of the subset is determined by $(width_minus1[4:0] + 1)$ (1..32). Instruction encoded in CI instruction format.

Decode Variables

```
Bits<5> width_minus1 = $encoding[6:2];
Bits<5> rd = $encoding[11:7];
```

Operation

```
XReg width = width_minus1 + 1;
X[rd] = (X[rd] >> 0) & ((1 << width) - 1);
```

Included in

Extension	Version
Xqci	~> 0.1.0
Xqcibm	~> 0.1.0

6.19. qc.c.mienter.nest

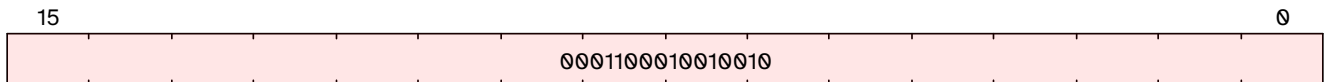
Synopsis

Machine mode interrupt enter

Mnemonic

```
qc.c.mienter.nest
```

Encoding



Description

Machine mode interrupt enter, interrupt nesting is enabled. Interrupt frame is saved in the stack. Interrupts are enabled. Instruction encoded in CI instruction format.

Decode Variables

qc.c.mienter.nest has no decode variables.

Operation

```
XReg virtual_address = get_and_validate_stack_pointer(X[2], $encoding);
XReg mepc_val = CSR[mepc].sw_read();
XReg mnepc_val = CSR[mnepc].sw_read();
XReg qc_mcause_val = CSR[qc.mcause].sw_read();
XReg reserved_val = 0;
if (CSR[mnstatus].NMIE == 1'b1) {
    write_memory<32>(virtual_address - 4, mepc_val, $encoding);
} else {
    write_memory<32>(virtual_address - 4, mnepc_val, $encoding);
}
write_memory<32>(virtual_address - 8, X[8][31:0], $encoding);
write_memory<32>(virtual_address - 12, qc_mcause_val, $encoding);
write_memory<32>(virtual_address - 16, X[1][31:0], $encoding);
write_memory<32>(virtual_address - 20, reserved_val, $encoding);
write_memory<32>(virtual_address - 24, X[5][31:0], $encoding);
write_memory<32>(virtual_address - 28, X[6][31:0], $encoding);
write_memory<32>(virtual_address - 32, X[7][31:0], $encoding);
write_memory<32>(virtual_address - 36, X[10][31:0], $encoding);
write_memory<32>(virtual_address - 40, X[11][31:0], $encoding);
write_memory<32>(virtual_address - 44, X[12][31:0], $encoding);
write_memory<32>(virtual_address - 48, X[13][31:0], $encoding);
write_memory<32>(virtual_address - 52, X[14][31:0], $encoding);
write_memory<32>(virtual_address - 56, X[15][31:0], $encoding);
write_memory<32>(virtual_address - 60, X[16][31:0], $encoding);
write_memory<32>(virtual_address - 64, X[17][31:0], $encoding);
write_memory<32>(virtual_address - 68, X[28][31:0], $encoding);
```

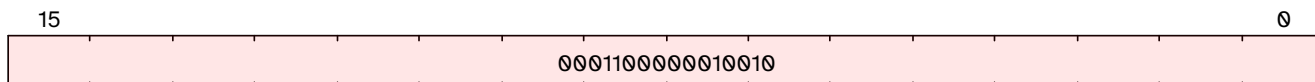
```
write_memory<32>(virtual_address - 72, X[29][31:0], $encoding);
write_memory<32>(virtual_address - 76, X[30][31:0], $encoding);
write_memory<32>(virtual_address - 80, X[31][31:0], $encoding);
X[8] = X[2];
X[2] = X[2] - 96;
CSR[mstatus].MIE = 1'b1;
CSR[qc.mcause].sw_write(qc_mcause_val | (1 << 26));
```

Included in

Extension	Version
Xqci	~> 0.1.0
Xqciint	~> 0.1.0

Machine mode interrupt enter

```
qc.c.minter
```



Machine mode interrupt enter, interrupt nesting is disabled. Interrupt frame is saved in the stack. Interrupts are disabled. Instruction encoded in CI instruction format.

qc.c.mienter has no decode variables.

```
XReg virtual_address = get_and_validate_stack_pointer(X[2], $encoding);
XReg mepc_val = CSR[mepc].sw_read();
XReg mnepc_val = CSR[mnepc].sw_read();
XReg qc_mcause_val = CSR[qc.mcause].sw_read();
XReg reserved_val = 0;
if (CSR[mnstatus].NMIE == 1'b1) {
    write_memory<32>(virtual_address - 4, mepc_val, $encoding);
} else {
    write_memory<32>(virtual_address - 4, mnepc_val, $encoding);
}
write_memory<32>(virtual_address - 8, X[8][31:0], $encoding);
write_memory<32>(virtual_address - 12, qc_mcause_val, $encoding);
write_memory<32>(virtual_address - 16, X[1][31:0], $encoding);
write_memory<32>(virtual_address - 20, reserved_val, $encoding);
write_memory<32>(virtual_address - 24, X[5][31:0], $encoding);
write_memory<32>(virtual_address - 28, X[6][31:0], $encoding);
write_memory<32>(virtual_address - 32, X[7][31:0], $encoding);
write_memory<32>(virtual_address - 36, X[10][31:0], $encoding);
write_memory<32>(virtual_address - 40, X[11][31:0], $encoding);
write_memory<32>(virtual_address - 44, X[12][31:0], $encoding);
write_memory<32>(virtual_address - 48, X[13][31:0], $encoding);
write_memory<32>(virtual_address - 52, X[14][31:0], $encoding);
write_memory<32>(virtual_address - 56, X[15][31:0], $encoding);
write_memory<32>(virtual_address - 60, X[16][31:0], $encoding);
write_memory<32>(virtual_address - 64, X[17][31:0], $encoding);
write_memory<32>(virtual_address - 68, X[28][31:0], $encoding);
```

```

write_memory<32>(virtual_address - 72, X[29][31:0], $encoding);
write_memory<32>(virtual_address - 76, X[30][31:0], $encoding);
write_memory<32>(virtual_address - 80, X[31][31:0], $encoding);
X[8] = X[2];
X[2] = X[2] - 96;

```

Included in

Extension	Version
Xqci	~> 0.1.0
Xqciint	~> 0.1.0

6.21. qc.c.mileaveret

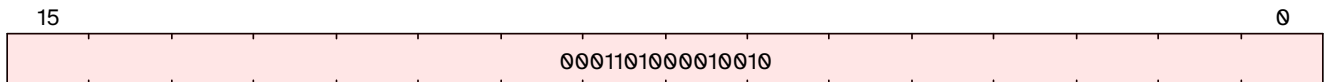
Synopsis

Machine mode interrupt exit

Mnemonic

```
qc.c.mileaveret
```

Encoding



Description

Machine mode interrupt exit. Interrupt frame is restored from the stack. Instruction encoded in CI instruction format.

Decode Variables

qc.c.mileaveret has no decode variables.

Operation

```
XReg virtual_address_sp = get_and_validate_stack_pointer(X[2], $encoding);
XReg virtual_address = virtual_address_sp + 96;
XReg prev_retpc = read_memory<32>(virtual_address - 4, $encoding);
XReg qc_mcause_val = read_memory<32>(virtual_address - 12, $encoding);
Bits<1> nmie_val = CSR[mnstatus].NMIE;
X[8] = read_memory<32>(virtual_address - 8, $encoding);
CSR[qc.mcause].sw_write(qc_mcause_val);
X[1] = read_memory<32>(virtual_address - 16, $encoding);
X[5] = read_memory<32>(virtual_address - 24, $encoding);
X[6] = read_memory<32>(virtual_address - 28, $encoding);
X[7] = read_memory<32>(virtual_address - 32, $encoding);
X[10] = read_memory<32>(virtual_address - 36, $encoding);
X[11] = read_memory<32>(virtual_address - 40, $encoding);
X[12] = read_memory<32>(virtual_address - 44, $encoding);
X[13] = read_memory<32>(virtual_address - 48, $encoding);
X[14] = read_memory<32>(virtual_address - 52, $encoding);
X[15] = read_memory<32>(virtual_address - 56, $encoding);
X[16] = read_memory<32>(virtual_address - 60, $encoding);
X[17] = read_memory<32>(virtual_address - 64, $encoding);
X[28] = read_memory<32>(virtual_address - 68, $encoding);
X[29] = read_memory<32>(virtual_address - 72, $encoding);
X[30] = read_memory<32>(virtual_address - 76, $encoding);
X[31] = read_memory<32>(virtual_address - 80, $encoding);
X[2] = X[2] + 96;
if (nmie_val == 1'b1) {
    XReg qc_mcause_val_masked = qc_mcause_val & ~(1 << 26) & ~(1 << 27) & ~(1 <<
```

```

29) & ~(0xFF << 12);
Bits<1> mpie_val = (qc_mcause_val >> 27) & 1;
Bits<1> mpdt_val = (qc_mcause_val >> 29) & 1;
Bits<4> mpil_val = (qc_mcause_val >> 16) & 0xF;
CSR[mstatus].MIE = mpie_val;
CSR[mstatus].MPIE = 1'b1;
if (implemented?(ExtensionName::Smdbltrp)) {
    CSR[mstatush].MDT = mpdt_val;
}
CSR[qc.mcause].sw_write(qc_mcause_val_masked | (1 << 27) | (mpie_val << 26) |
(0 << 29) | (mpil_val << 12) | (0xF << 16));
if (mpdt_val == 1'b0) {
    if (CSR[mstatus].MPP != 2'b11) {
        CSR[mstatus].MPRV = 0;
    }
    if (CSR[mstatus].MPP == 2'b00) {
        set_mode(PrivilegeMode::U);
    } else if (CSR[mstatus].MPP == 2'b01) {
        set_mode(PrivilegeMode::S);
    } else if (CSR[mstatus].MPP == 2'b11) {
        set_mode(PrivilegeMode::M);
    }
    CSR[mstatus].MPP = implemented?(ExtensionName::U) ? 2'b00 : 2'b11;
}
$pc = CSR[mepc].sw_read();
} else {
    XReg qc_mcause_val_masked = qc_mcause_val & ~(1 << 26) & ~(1 << 28) & ~(1 <<
30) & ~(0xF << 12) & ~(0xF << 20);
Bits<1> mnpie_val = (qc_mcause_val >> 28) & 1;
Bits<4> mnpil_val = (qc_mcause_val >> 20) & 0xF;
CSR[mnstatus].MIE = mnpie_val;
CSR[mnstatus].NMIE = 1'b1;
CSR[qc.mcause].sw_write(qc_mcause_val_masked | (1 << 28) | (mnpie_val << 26)
| (1 << 30) | (mnpil_val << 12) | (0xF << 20));
if (CSR[mnstatus].MNPP != 2'b11) {
    CSR[mnstatus].MPRV = 0;
    if (implemented?(ExtensionName::Smdbltrp)) {
        CSR[mnstatush].MDT = 1'b0;
    }
}
if (CSR[mnstatus].MNPP == 2'b00) {
    set_mode(PrivilegeMode::U);
} else if (CSR[mnstatus].MNPP == 2'b01) {
    set_mode(PrivilegeMode::S);
} else if (CSR[mnstatus].MNPP == 2'b11) {
    set_mode(PrivilegeMode::M);
}
CSR[mnstatus].MNPP = implemented?(ExtensionName::U) ? 2'b00 : 2'b11;
$pc = CSR[mnepc].sw_read();
}

```

Included in

Extension	Version
Xqci	~> 0.1.0
Xqciint	~> 0.1.0

6.22. qc.c.mnret

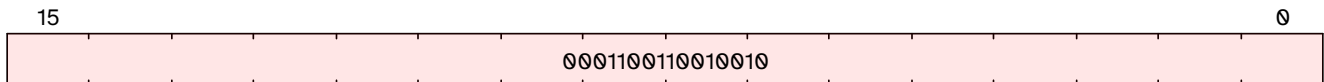
Synopsis

Machine NMI Return

Mnemonic

```
qc.c.mnret
```

Encoding



Description

Returns from an NMI in M-mode. Instruction encoded in CI instruction format.

Decode Variables

qc.c.mnret has no decode variables.

Operation

```
XReg qc_mcause_val = CSR[qc.mcause].sw_read();
XReg qc_mcause_val_masked = qc_mcause_val & ~(1 << 26) & ~(1 << 28) & ~(1 <<
30) & ~(0xF << 12) & ~(0xF << 20);
Bits<1> mnpie_val = (qc_mcause_val >> 28) & 1;
Bits<4> mnpil_val = (qc_mcause_val >> 20) & 0xF;
CSR[mstatus].MIE = mnpie_val;
CSR[mnstatus].NMIE = 1'b1;
CSR[qc.mcause].sw_write(qc_mcause_val_masked | (1 << 28) | (mnpie_val << 26) |
(1 << 30) | (mnpil_val << 12) | (0xF << 20));
if (CSR[mnstatus].MNPP != 2'b11) {
    CSR[mstatus].MPRV = 0;
    if (implemented?(ExtensionName::Smdbltrp)) {
        CSR[mstatus].MDT = 1'b0;
    }
}
if (CSR[mnstatus].MNPP == 2'b00) {
    set_mode(PrivilegeMode::U);
} else if (CSR[mnstatus].MNPP == 2'b01) {
    set_mode(PrivilegeMode::S);
} else if (CSR[mnstatus].MNPP == 2'b11) {
    set_mode(PrivilegeMode::M);
}
CSR[mnstatus].MNPP = implemented?(ExtensionName::U) ? 2'b00 : 2'b11;
$pc = CSR[mnepc].sw_read();
```

Included in

Extension	Version
Xqci	~> 0.1.0
Xqciint	~> 0.1.0

6.23. qc.c.mret

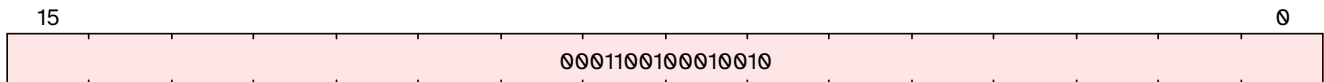
Synopsis

Machine Exception Return

Mnemonic

qc.c.mret

Encoding



Description

Returns from an exception in M-mode. Instruction encoded in CI instruction format.

Decode Variables

qc.c.mret has no decode variables.

Operation

```
XReg qc_mcause_val = CSR[qc.mcause].sw_read();
XReg qc_mcause_val_masked = qc_mcause_val & ~(1 << 26) & ~(1 << 27) & ~(1 <<
29) & ~(0xFF << 12);
Bits<1> mpie_val = (qc_mcause_val >> 27) & 1;
Bits<1> mpdt_val = (qc_mcause_val >> 29) & 1;
Bits<4> mpil_val = (qc_mcause_val >> 16) & 0xF;
CSR[mstatus].MIE = mpie_val;
CSR[mstatus].MPIE = 1'b1;
if (implemented?(ExtensionName::Smdbltrp)) {
    CSR[mstatush].MDT = mpdt_val;
}
CSR[qc.mcause].sw_write(qc_mcause_val_masked | (1 << 27) | (mpie_val << 26) |
(0 << 29) | (mpil_val << 12) | (0xF << 16));
if (mpdt_val == 1'b0) {
    if (CSR[mstatus].MPP != 2'b11) {
        CSR[mstatus].MPRV = 0;
    }
    if (CSR[mstatus].MPP == 2'b00) {
        set_mode(PrivilegeMode::U);
    } else if (CSR[mstatus].MPP == 2'b01) {
        set_mode(PrivilegeMode::S);
    } else if (CSR[mstatus].MPP == 2'b11) {
        set_mode(PrivilegeMode::M);
    }
    CSR[mstatus].MPP = implemented?(ExtensionName::U) ? 2'b00 : 2'b11;
}
```

```
$pc = CSR[mepc].sw_read();
```

Included in

Extension	Version
Xqci	~> 0.1.0
Xqciint	~> 0.1.0

6.24. qc.c.muliadd

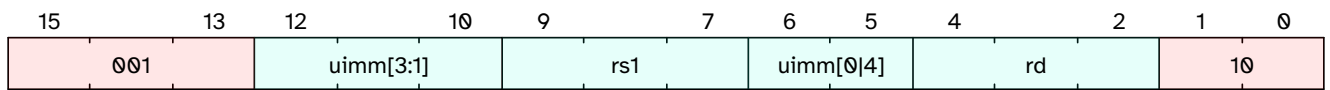
Synopsis

Multiply and accumulate (Immediate)

Mnemonic

```
qc.c.muliadd rd, rs1, uimm
```

Encoding



Description

Increments `rd` by the multiplication of `rs1` and an unsigned immediate Instruction encoded in CL instruction format.

Decode Variables

```
Bits<5> uimm = { $encoding[5], $encoding[12:10], $encoding[6] };
Bits<3> rs1 = $encoding[9:7];
Bits<3> rd = $encoding[4:2];
```

Operation

```
XReg reg = creg2reg(rd);
XReg src = creg2reg(rs1);
XReg orig_val = X[reg];
X[reg] = orig_val + (X[src] * uimm);
```

Included in

Extension	Version
Xqci	~> 0.1.0
Xqciac	~> 0.1.0

6.25. qc.c.mveqz

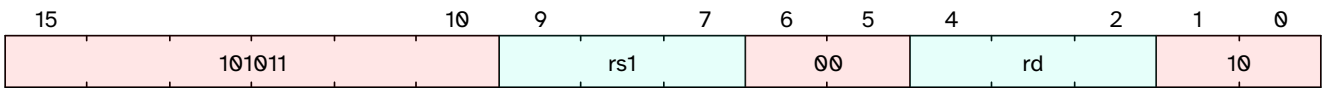
Synopsis

Conditional Move if equal to zero

Mnemonic

```
qc.c.mveqz  rd, rs1
```

Encoding



Description

Move rs1 to rd if rd == 0, keep rd value otherwise Instruction encoded in CL instruction format.

Decode Variables

```
Bits<3> rs1 = $encoding[9:7];
Bits<3> rd = $encoding[4:2];
```

Operation

```
XReg reg = creg2reg(rd);
XReg src = creg2reg(rs1);
XReg orig_val = X[reg];
X[reg] = (orig_val == 0) ? X[src] : orig_val;
```

Included in

Extension	Version
Xqci	~> 0.1.0
Xqcicm	~> 0.1.0

6.26. qc.c.pttrace

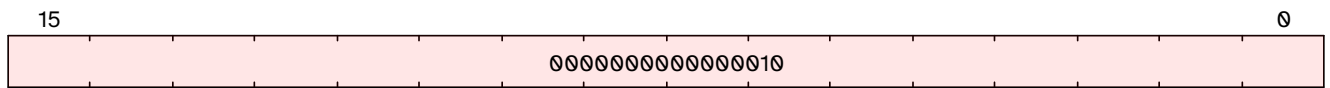
Synopsis

Tracing pseudo-instruction (hint) working only in simulation environment

Mnemonic

```
qc.c.pttrace
```

Encoding



Description

The tracing instruction have no explicit arguments. Implicit arguments defined by simulation environment implementation. Instruction is used to signal simulator to collect some tracing information. Instruction encoded in CI instruction format.

Decode Variables

qc.c.pttrace has no decode variables.

Operation

```
XReg func = 9;
XReg arg = 0;
iss_syscall(func, arg);
```

Included in

Extension	Version
Xqci	~> 0.1.0
Xqcisim	~> 0.1.0

6.27. qc.c.setint

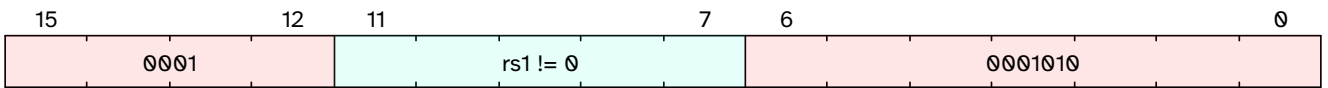
Synopsis

Set interrupt (Register)

Mnemonic

```
qc.c.setint  rs1
```

Encoding



Description

Set interrupt, interrupt number is in rs1. Instruction encoded in CI instruction format.

Decode Variables

```
Bits<5> rs1 = $encoding[11:7];
```

Operation

```
Bits<12> MCLICIP0_ADDR = CSR[qc.mclicip0].address();
XReg idx = rs1 / 32;
XReg bit = rs1 % 32;
XReg pre_csr = CSR[MCLICIP0_ADDR + idx].sw_read();
CSR[MCLICIP0_ADDR + idx].sw_write(pre_csr | (1 << bit));
```

Included in

Extension	Version
Xqci	~> 0.1.0
Xqciint	~> 0.1.0

6.28. qc.c.sync

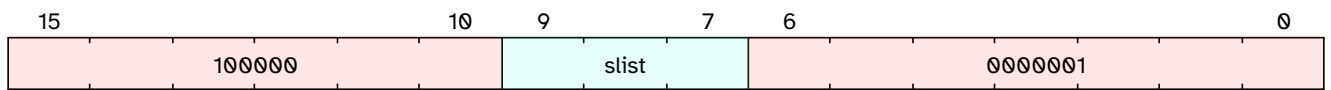
Synopsis

Non-memory-mapped device read-after-write synchronization to completion

Mnemonic

```
qc.c.sync  slist
```

Encoding



Description

This synchronization instruction delays execution till last read (input) from non-memory-mapped device transactions are completed for specified devices. As well, instruction implies fence.tso functionality for all memory accesses. Immediate argument is 3-bit encoding of 5-bit bitmask field that specifies up to five pre-defined devices. Values of bitmask encoding supported: 0,1,2,4,8,16,15,31 Instruction encoded in CB instruction format.

Decode Variables

```
Bits<3> slist = $encoding[9:7];
```

Operation

```
Bits<5> bitmask;
if (slist == 0) {
    bitmask = 0;
} else if (slist < 6) {
    XReg shift = slist - 1;
    bitmask = (1 << shift);
} else {
    XReg shift = slist - 2;
    bitmask = (1 << shift) - 1;
}
fence_tso();
sync_read_after_write_device(true, bitmask);
```

Included in

Extension	Version
Xqci	~> 0.1.0
Xqcisync	~> 0.1.0

6.29. qc.c.syncr

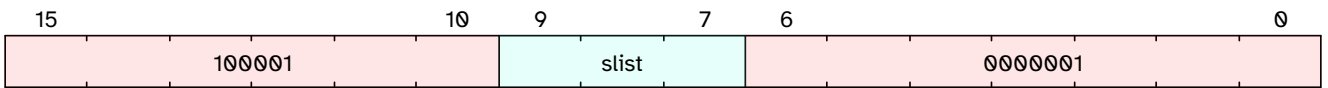
Synopsis

Non-memory-mapped device read-after-write synchronization

Mnemonic

```
qc.c.syncr  slist
```

Encoding



Description

This synchronization instruction delays execution till last read (input) from non-memory-mapped device transactions are started for specified devices. As well, instruction implies fence.tso functionality for all memory accesses. Immediate argument is 3-bit encoding of 5-bit bitmask field that specifies up to five pre-defined devices. Values of bitmask encoding supported: 0,1,2,4,8,16,15,31 Instruction encoded in CB instruction format.

Decode Variables

```
Bits<3> slist = $encoding[9:7];
```

Operation

```
Bits<5> bitmask;
if (slist == 0) {
    bitmask = 0;
} else if (slist < 6) {
    XReg shift = slist - 1;
    bitmask = (1 << shift);
} else {
    XReg shift = slist - 2;
    bitmask = (1 << shift) - 1;
}
fence_tso();
sync_read_after_write_device(false, bitmask);
```

Included in

Extension	Version
Xqci	~> 0.1.0
Xqcisync	~> 0.1.0

6.30. qc.c.syncwf

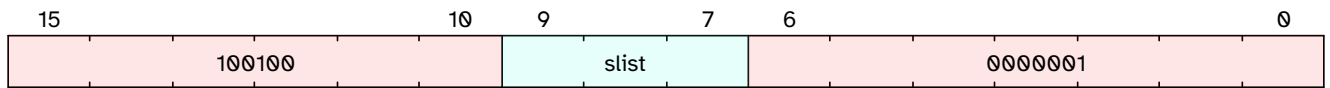
Synopsis

Non-memory-mapped device write-after-read synchronization

Mnemonic

```
qc.c.syncwf  slist
```

Encoding



Description

This synchronization instruction delays execution till last write (output) to non-memory-mapped device transactions are started for specified devices. As well, instruction implies fence.tso functionality for all memory accesses. Immediate argument is 3-bit encoding of 5-bit bitmask field that specifies up to five pre-defined devices. Values of bitmask encoding supported: 0,1,2,4,8,16,31 Instruction encoded in CB instruction format.

Decode Variables

```
Bits<3> slist = $encoding[9:7];
```

Operation

```
Bits<5> bitmask;
if (slist == 0) {
    bitmask = 0;
} else if (slist < 6) {
    XReg shift = slist - 1;
    bitmask = (1 << shift);
} else {
    XReg shift = slist - 2;
    bitmask = (1 << shift) - 1;
}
fence_tso();
sync_write_after_read_device(false, bitmask);
```

Included in

Extension	Version
Xqci	~> 0.1.0
Xqcisync	~> 0.1.0

6.31. qc.c.syncwl

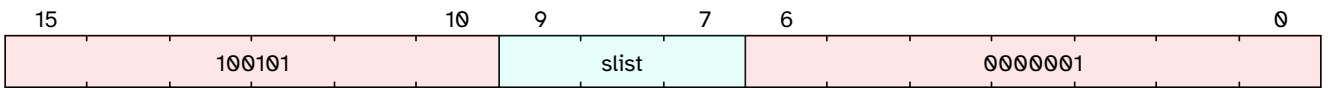
Synopsis

Non-memory-mapped device write-after-read synchronization to completion

Mnemonic

```
qc.c.syncwl  slist
```

Encoding



Description

This synchronization instruction delays execution till last write (output) to non-memory-mapped device transactions are completed for specified devices. As well, instruction implies fence.tso functionality for all memory accesses. Immediate argument is 3-bit encoding of 5-bit bitmask field that specifies up to five pre-defined devices. Values of bitmask encoding supported: 0,1,2,4,8,16,15,31 Instruction encoded in CB instruction format.

Decode Variables

```
Bits<3> slist = $encoding[9:7];
```

Operation

```
Bits<5> bitmask;
if (slist == 0) {
    bitmask = 0;
} else if (slist < 6) {
    XReg shift = slist - 1;
    bitmask = (1 << shift);
} else {
    XReg shift = slist - 2;
    bitmask = (1 << shift) - 1;
}
fence_tso();
sync_write_after_read_device(true, bitmask);
```

Included in

Extension	Version
Xqci	~> 0.1.0
Xqcisync	~> 0.1.0

6.32. qc.clo

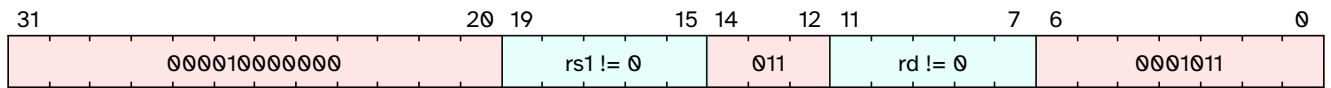
Synopsis

Count leading ones

Mnemonic

```
qc.clo rd, rs1
```

Encoding



Description

Count the number of ones before the first zero in `rs1`, starting from the MSB and progressing to the LSB. Accordingly, if the input is `~0`, the output is `XLEN`, and if the most-significant bit of the input is a `0`, the output is `0`. output written to the `rd` Instruction encoded in I instruction format.

Decode Variables

```
Bits<5> rs1 = $encoding[19:15];
Bits<5> rd = $encoding[11:7];
```

Operation

```
X[rd] = (xlen() - 1) - $signed(highest_set_bit(~X[rs1]));
```

Included in

Extension	Version
Xqci	~> 0.1.0
Xqcibm	~> 0.1.0

6.33. qc.clrinti

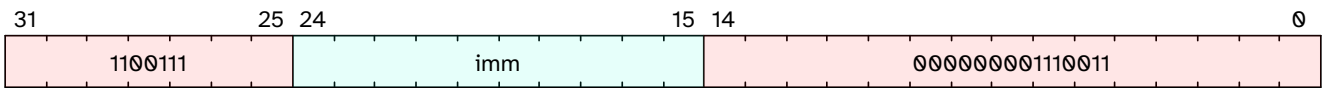
Synopsis

Clear interrupt (Immediate)

Mnemonic

```
qc.clrinti imm
```

Encoding



Description

Clear interrupt, interrupt number is in `imm` (0 - 1023). Instruction encoded in R instruction format.

Decode Variables

```
Bits<10> imm = $encoding[24:15];
```

Operation

```
Bits<12> MCLICIP0_ADDR = CSR[qc.mclicip0].address();
XReg idx = imm / 32;
XReg bit = imm % 32;
XReg pre_csr = CSR[MCLICIP0_ADDR + idx].sw_read();
CSR[MCLICIP0_ADDR + idx].sw_write(pre_csr & ~(1 << bit));
```

Included in

Extension	Version
Xqci	~> 0.1.0
Xqciint	~> 0.1.0

6.34. qc.compress2

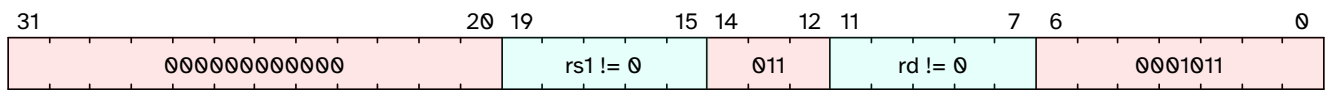
Synopsis

Bit compression (every 2nd bit)

Mnemonic

```
qc.compress2 rd, rs1
```

Encoding



Description

Bit compression (every 2nd bit) of `rs1`, zero-pad bits [31:16] of the result. Write result to `rd`. Instruction encoded in I instruction format.

Decode Variables

```
Bits<5> rs1 = $encoding[19:15];
Bits<5> rd = $encoding[11:7];
```

Operation

```
XReg b0 = {X[rs1][14], X[rs1][12], X[rs1][10], X[rs1][8], X[rs1][6], X[rs1][4],
X[rs1][2], X[rs1][0]};
XReg b1 = {X[rs1][30], X[rs1][28], X[rs1][26], X[rs1][24], X[rs1][22], X[rs1][20],
X[rs1][18], X[rs1][16]};
X[rd] = {16'b0, b1[7:0], b0[7:0]};
```

Included in

Extension	Version
Xqci	~> 0.1.0
Xqcibm	~> 0.1.0

6.35. qc.compress3

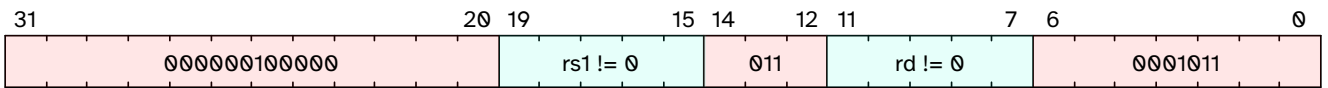
Synopsis

Bit compression (every 3rd bit)

Mnemonic

```
qc.compress3 rd, rs1
```

Encoding



Description

Bit compression (every 3rd bit) of `rs1`, zero-pad bits [31:11] of the result. Write result to `rd`. Instruction encoded in I instruction format.

Decode Variables

```
Bits<5> rs1 = $encoding[19:15];
Bits<5> rd = $encoding[11:7];
```

Operation

```
XReg b0 = {X[rs1][21], X[rs1][18], X[rs1][15], X[rs1][12], X[rs1][9], X[rs1][6], X[rs1][3], X[rs1][0]};
XReg b1 = {5'b0, X[rs1][30], X[rs1][27], X[rs1][24]};
X[rd] = {21'b0, b1[2:0], b0[7:0]};
```

Included in

Extension	Version
Xqci	~> 0.1.0
Xqcibm	~> 0.1.0

6.36. qc.csrrwr

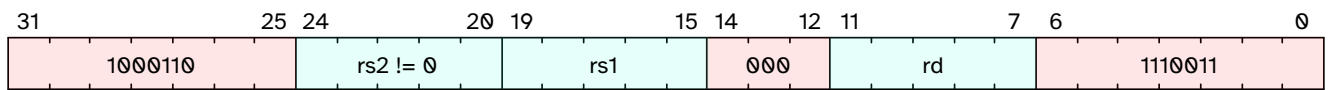
Synopsis

Atomic Read/Write CSR (Register)

Mnemonic

```
qc.csrrwr rd, rs1, rs2
```

Encoding



Description

Atomically swap values in the CSRs and integer registers. Read the old value of the CSR, zero-extends the value to XLEN bits, and then write it to integer register `rd`. The CSR number is in `rs2` register. The initial value in `rs1` is written to the CSR. If `rd=x0`, then the instruction shall not read the CSR and shall not cause any of the side effects that might occur on a CSR read. Instruction encoded in R instruction format.

Decode Variables

```
Bits<5> rs2 = $encoding[24:20];
Bits<5> rs1 = $encoding[19:15];
Bits<5> rd = $encoding[11:7];
```

Operation

```
XReg csr = X[rs2];
if (rd != 0) {
    X[rd] = CSR[csr].sw_read();
}
CSR[csr].sw_write(X[rs1]);
```

Included in

Extension	Version
Xqci	~> 0.1.0
Xqcicr	~> 0.1.0

6.37. qc.csrrwri

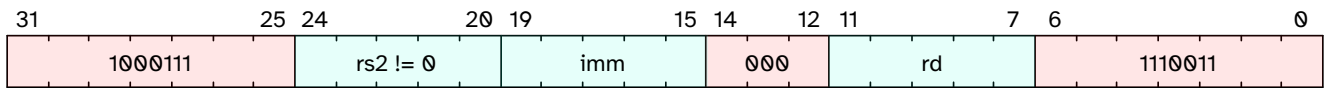
Synopsis

Atomic Read/Write CSR (Register) Immediate

Mnemonic

```
qc.csrrwri rd, imm, rs2
```

Encoding



Description

Atomically write CSR using a 5-bit immediate `imm`, and load the previous value into `rd`. Read the old value of the CSR, zero-extends the value to `XLEN` bits, and then write it to integer register `rd`. The CSR number is in `rs2` register. The 5-bit `uimm` field is zero-extended and written to the CSR. If `rd` $\neq$ `x0`, then the instruction shall not read the CSR and shall not cause any of the side effects that might occur on a CSR read. Instruction encoded in R instruction format.

Decode Variables

```
Bits<5> rs2 = $encoding[24:20];
Bits<5> imm = $encoding[19:15];
Bits<5> rd = $encoding[11:7];
```

Operation

```
XReg csr = X[rs2];
if (rd != 0) {
    X[rd] = CSR[csr].sw_read();
}
CSR[csr].sw_write({{MXLEN - 5{1'b0}}}, imm);
```

Included in

Extension	Version
Xqci	~> 0.1.0
Xqcicsr	~> 0.1.0

6.38. qc.cto

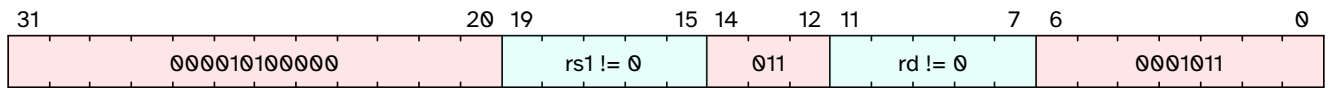
Synopsis

Count trailing ones

Mnemonic

```
qc.cto rd, rs1
```

Encoding



Description

Count the number of ones before the first zero in `rs1`, starting from the LSB and progressing to the MSB. Accordingly, if the input is `~0`, the output is `XLEN`, and if the least-significant bit of the input is a `0`, the output is `0`. Output written to `rd`. Instruction encoded in I instruction format.

Decode Variables

```
Bits<5> rs1 = $encoding[19:15];
Bits<5> rd = $encoding[11:7];
```

Operation

```
X[rd] = lowest_set_bit(~X[rs1]);
```

Included in

Extension	Version
Xqci	~> 0.1.0
Xqcibm	~> 0.1.0

6.39. qc.e.addai

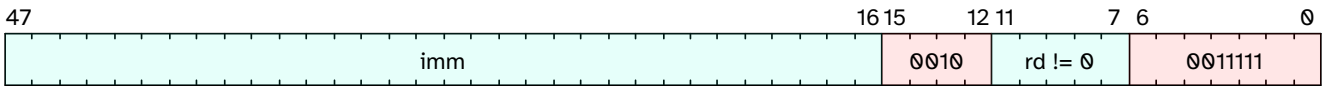
Synopsis

Add immediate

Mnemonic

```
qc.e.addai rd, imm
```

Encoding



Description

Add a 32-bit immediate `imm` to the value in `rd`, and store the result back in `rd`. Instruction encoded in QC.EAI instruction format.

Decode Variables

```
Bits<32> imm = $encoding[47:16];
Bits<5> rd = $encoding[11:7];
```

Operation

```
X[rd] = X[rd] + imm;
```

Included in

Extension	Version
Xqci	~> 0.1.0
Xqcilia	~> 0.1.0

6.40. qc.e.addi

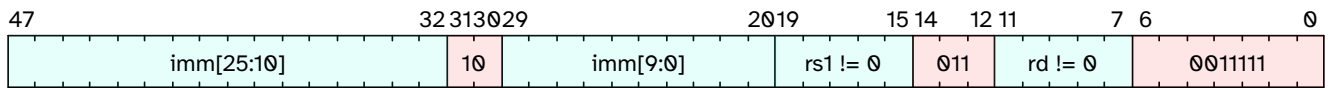
Synopsis

Add immediate

Mnemonic

```
qc.e.addi rd, rs1, imm
```

Encoding



Description

Add a sign-extended 26-bit immediate `imm` to the value in `rs1`, and store the result in `rd`. Instruction encoded in QC.EI instruction format.

Decode Variables

```
Bits<26> imm = { $encoding[47:32], $encoding[29:20] };
Bits<5> rs1 = $encoding[19:15];
Bits<5> rd = $encoding[11:7];
```

Operation

```
X[rd] = X[rs1] + sext(imm, 26);
```

Included in

Extension	Version
Xqci	~> 0.1.0
Xqcilia	~> 0.1.0

6.41. qc.e.andai

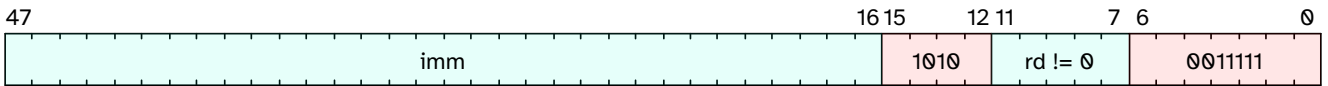
Synopsis

And immediate

Mnemonic

```
qc.e.andai rd, imm
```

Encoding



Description

And a 32-bit immediate `imm` to the value in `rd`, and store the result back in `rd`. Instruction encoded in QC.EAI instruction format.

Decode Variables

```
Bits<32> imm = $encoding[47:16];
Bits<5> rd = $encoding[11:7];
```

Operation

```
X[rd] = X[rd] & imm;
```

Included in

Extension	Version
Xqci	~> 0.1.0
Xqcilia	~> 0.1.0

6.42. qc.e.andi

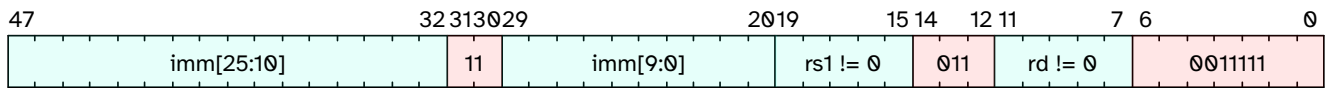
Synopsis

Add immediate

Mnemonic

```
qc.e.andi rd, rs1, imm
```

Encoding



Description

And a sign-extended 26-bit immediate `imm` to the value in `rs1`, and store the result in `rd`. Instruction encoded in QC.EI instruction format.

Decode Variables

```
Bits<26> imm = { $\$encoding[47:32]$ ,  $\$encoding[29:20]$ };  
Bits<5> rs1 =  $\$encoding[19:15]$ ;  
Bits<5> rd =  $\$encoding[11:7]$ ;
```

Operation

```
X[rd] = X[rs1] & sext(imm, 26);
```

Included in

Extension	Version
Xqci	~> 0.1.0
Xqcilia	~> 0.1.0

6.43. qc.e.beqi

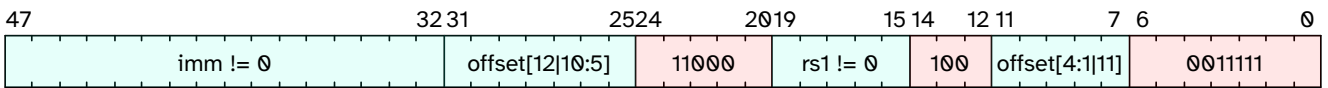
Synopsis

Branch on equal (immediate)

Mnemonic

```
qc.e.beqi  rs1, imm, offset
```

Encoding



Description

Branches to PC + offset if the value in rs1 is equal to the signed immediate imm Instruction encoded in QC.EB instruction format.

Decode Variables

```
signed Bits<13> offset = sext({$encoding[31], $encoding[7], $encoding[30:25],
$encoding[11:8], 1'd0});
Bits<16> imm = $encoding[47:32];
Bits<5> rs1 = $encoding[19:15];
```

Operation

```
if ($signed(X[rs1]) == $signed(imm)) {
    jump_halfword($pc + $signed(offset));
}
```

Included in

Extension	Version
Xqci	~> 0.1.0
Xqcibi	~> 0.1.0

6.44. qc.e.bgei

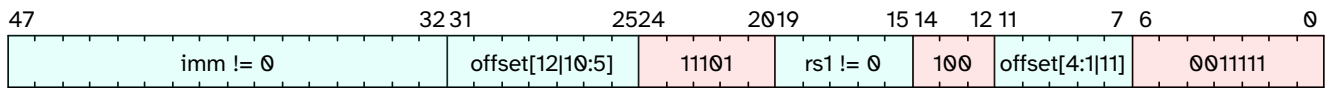
Synopsis

Branch on greater than or equal (immediate)

Mnemonic

```
qc.e.bgei  rs1, imm, offset
```

Encoding



Description

Branches to PC + offset if the value in rs1 is greater than or equal to the immediate imm. Instruction encoded in QC.EB instruction format.

Decode Variables

```
signed Bits<13> offset = sext({$encoding[31], $encoding[7], $encoding[30:25],  
$encoding[11:8], 1'd0});  
Bits<16> imm = $encoding[47:32];  
Bits<5> rs1 = $encoding[19:15];
```

Operation

```
if ($signed(X[rs1]) >= $signed(imm)) {  
    jump_halfword($pc + $signed(offset));  
}
```

Included in

Extension	Version
Xqci	~> 0.1.0
Xqcibi	~> 0.1.0

6.45. qc.e.bgeui

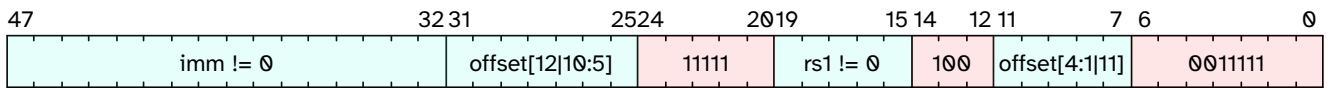
Synopsis

Branch on greater than or equal unsigned (immediate)

Mnemonic

```
qc.e.bgeui  rs1, imm, offset
```

Encoding



Description

Branches to PC + offset if the unsigned value in rs1 is greater than or equal to the unsigned immediate imm. Instruction encoded in QC.EB instruction format.

Decode Variables

```
signed Bits<13> offset = sext({$encoding[31], $encoding[7], $encoding[30:25],
$encoding[11:8], 1'd0});
Bits<16> imm = $encoding[47:32];
Bits<5> rs1 = $encoding[19:15];
```

Operation

```
if (X[rs1] >= imm) {
    jump_halfword($pc + $signed(offset));
}
```

Included in

Extension	Version
Xqci	~> 0.1.0
Xqcibi	~> 0.1.0

6.46. qc.e.blti

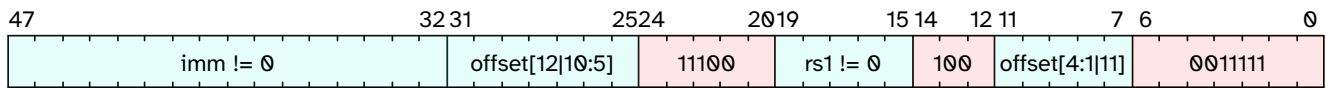
Synopsis

Branch on less than (immediate)

Mnemonic

```
qc.e.blti  rs1, imm, offset
```

Encoding



Description

Branches to PC + offset if the value in rs1 is less than the immediate imm. Instruction encoded in QC.EB instruction format.

Decode Variables

```
signed Bits<13> offset = sext({$encoding[31], $encoding[7], $encoding[30:25],
$encoding[11:8], 1'd0});
Bits<16> imm = $encoding[47:32];
Bits<5> rs1 = $encoding[19:15];
```

Operation

```
if ($signed(X[rs1]) < $signed(imm)) {
    jump_halfword($pc + $signed(offset));
}
```

Included in

Extension	Version
Xqci	~> 0.1.0
Xqcibi	~> 0.1.0

6.47. qc.e.bltui

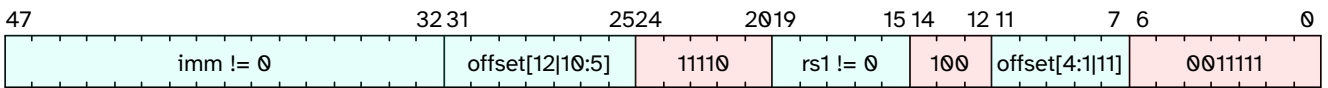
Synopsis

Branch on less than unsigned (immediate)

Mnemonic

```
qc.e.bltui  rs1, imm, offset
```

Encoding



Description

Branches to PC + offset if the unsigned value in rs1 is less than the unsigned immediate imm. Instruction encoded in QC.EB instruction format.

Decode Variables

```
signed Bits<13> offset = sext({$encoding[31], $encoding[7], $encoding[30:25],
$encoding[11:8], 1'd0});
Bits<16> imm = $encoding[47:32];
Bits<5> rs1 = $encoding[19:15];
```

Operation

```
if (X[rs1] < imm) {
    jump_halfword($pc + $signed(offset));
}
```

Included in

Extension	Version
Xqci	~> 0.1.0
Xqcibi	~> 0.1.0

6.48. qc.e.bnei

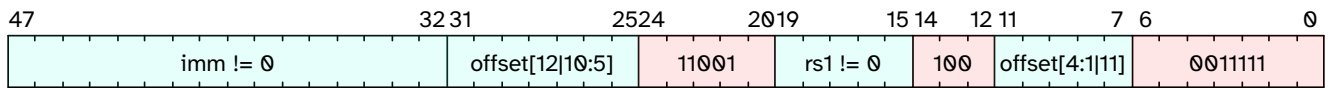
Synopsis

Branch on not equal (immediate)

Mnemonic

```
qc.e.bnei  rs1, imm, offset
```

Encoding



Description

Branches to PC + offset if the value in rs1 is not equal to the signed immediate imm. Instruction encoded in QC.EB instruction format.

Decode Variables

```
signed Bits<13> offset = sext({$encoding[31], $encoding[7], $encoding[30:25],
$encoding[11:8], 1'd0});
Bits<16> imm = $encoding[47:32];
Bits<5> rs1 = $encoding[19:15];
```

Operation

```
if ($signed(X[rs1]) != $signed(imm)) {
    jump_halfword($pc + $signed(offset));
}
```

Included in

Extension	Version
Xqci	~> 0.1.0
Xqcibi	~> 0.1.0

6.49. qc.e.j

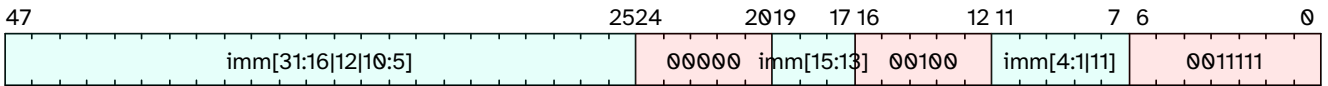
Synopsis

Jump

Mnemonic

```
qc.e.j  imm
```

Encoding



Description

Jump to a PC-relative offset. Instruction encoded in QC.EJ instruction format.

Decode Variables

```
signed Bits<32> imm = sext({$encoding[47:32], $encoding[19:17], $encoding[31],  
$encoding[7], $encoding[30:25], $encoding[11:8], 1'd0});
```

Operation

```
jump_halfword($pc + imm);
```

Included in

Extension	Version
Xqci	~> 0.1.0
Xqcilb	~> 0.1.0

6.50. qc.e.jal

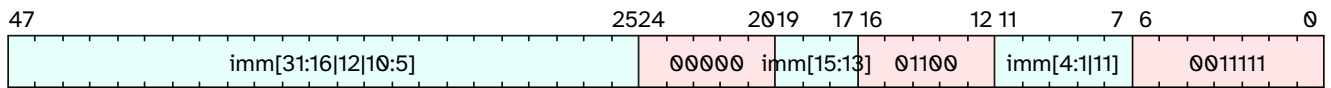
Synopsis

Jump and link

Mnemonic

```
qc.e.jal  imm
```

Encoding



Description

Jump to a PC-relative offset and store the return. Instruction encoded in QC.EJ instruction format. address in x1.

Decode Variables

```
signed Bits<32> imm = sext({$encoding[47:32], $encoding[19:17], $encoding[31],  
$encoding[7], $encoding[30:25], $encoding[11:8], 1'd0});
```

Operation

```
XReg retrun_addr = $pc + 6;  
jump_halfword($pc + imm);  
X[1] = retrun_addr;
```

Included in

Extension	Version
Xqci	~> 0.1.0
Xqcilb	~> 0.1.0

6.51. qc.e.lb

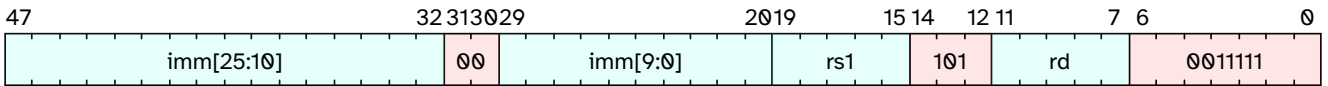
Synopsis

Load byte

Mnemonic

```
qc.e.lb rd, imm(rs1)
```

Encoding



Description

Load 8 bits of data into register `rd` from an address formed by adding `rs1` to a signed offset `imm`. Sign extend the result. Instruction encoded in QC.EI instruction format.

Decode Variables

```
Bits<26> imm = { $encoding[47:32], $encoding[29:20] };
Bits<5> rs1 = $encoding[19:15];
Bits<5> rd = $encoding[11:7];
```

Operation

```
XReg virtual_address = X[rs1] + imm;
X[rd] = sext(read_memory<8>(virtual_address, $encoding), 8);
```

Included in

Extension	Version
Xqci	~> 0.1.0
Xqcilo	~> 0.1.0

6.52. qc.e.lbu

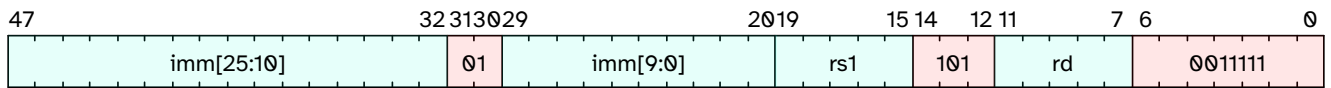
Synopsis

Load byte unsigned

Mnemonic

```
qc.e.lbu rd, imm(rs1)
```

Encoding



Description

Load 8 bits of data into register `rd` from an address formed by adding `rs1` to a signed offset `imm`. Zero extend the result. Instruction encoded in QC.EI instruction format.

Decode Variables

```
Bits<26> imm = { $encoding[47:32], $encoding[29:20] };
Bits<5> rs1 = $encoding[19:15];
Bits<5> rd = $encoding[11:7];
```

Operation

```
XReg virtual_address = X[rs1] + imm;
X[rd] = read_memory<8>(virtual_address, $encoding);
```

Included in

Extension	Version
Xqci	~> 0.1.0
Xqcilo	~> 0.1.0

6.53. qc.e.lh

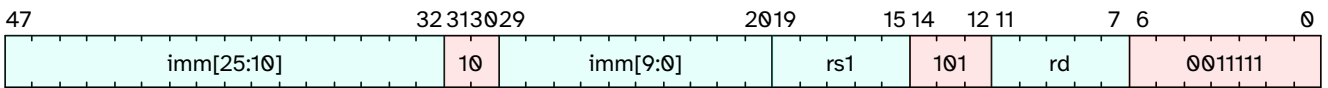
Synopsis

Load halfword

Mnemonic

```
qc.e.lh rd, imm(rs1)
```

Encoding



Description

Load 16 bits of data into register `rd` from an address formed by adding `rs1` to a signed offset `imm`. Sign extend the result. Instruction encoded in QC.EI instruction format.

Decode Variables

```
Bits<26> imm = { $encoding[47:32], $encoding[29:20] };
Bits<5> rs1 = $encoding[19:15];
Bits<5> rd = $encoding[11:7];
```

Operation

```
XReg virtual_address = X[rs1] + imm;
X[rd] = sext(read_memory<16>(virtual_address, $encoding), 16);
```

Included in

Extension	Version
Xqci	~> 0.1.0
Xqcilo	~> 0.1.0

6.54. qc.e.lhu

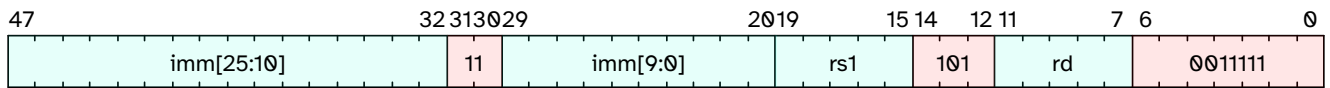
Synopsis

Load halfword unsigned

Mnemonic

```
qc.e.lhu rd, imm(rs1)
```

Encoding



Description

Load 16 bits of data into register `rd` from an address formed by adding `rs1` to a signed offset `imm`. Zero extend the result. Instruction encoded in QC.EI instruction format.

Decode Variables

```
Bits<26> imm = { $encoding[47:32], $encoding[29:20] };
Bits<5> rs1 = $encoding[19:15];
Bits<5> rd = $encoding[11:7];
```

Operation

```
XReg virtual_address = X[rs1] + imm;
X[rd] = read_memory<16>(virtual_address, $encoding);
```

Included in

Extension	Version
Xqci	~> 0.1.0
Xqcilo	~> 0.1.0

6.55. qc.e.li

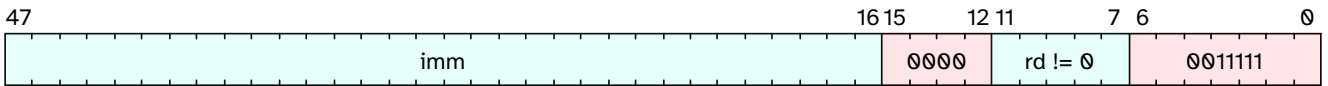
Synopsis

Load immediate large

Mnemonic

```
qc.e.li rd, imm
```

Encoding



Description

Loads the 32-bit immediate `imm` into `rd`. Instruction encoded in QC.EAI instruction format.

Decode Variables

```
Bits<32> imm = $encoding[47:16];
Bits<5> rd = $encoding[11:7];
```

Operation

```
X[rd] = imm;
```

Included in

Extension	Version
Xqci	~> 0.1.0
Xqcili	~> 0.1.0

6.56. qc.e.lw

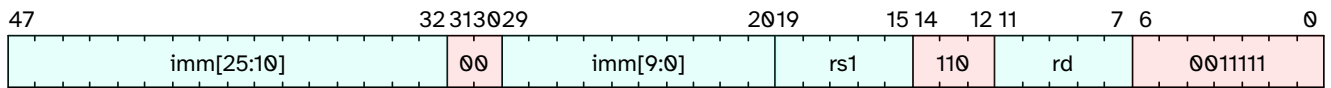
Synopsis

Load word

Mnemonic

```
qc.e.lw rd, imm(rs1)
```

Encoding



Description

Load 32 bits of data into register `rd` from an address formed by adding `rs1` to a signed offset `imm`. Sign extend the result. Instruction encoded in QC.EI instruction format.

Decode Variables

```
Bits<26> imm = { $encoding[47:32], $encoding[29:20] };
Bits<5> rs1 = $encoding[19:15];
Bits<5> rd = $encoding[11:7];
```

Operation

```
XReg virtual_address = X[rs1] + imm;
X[rd] = read_memory<32>(virtual_address, $encoding);
```

Included in

Extension	Version
Xqci	~> 0.1.0
Xqcilo	~> 0.1.0

6.57. qc.e.orai

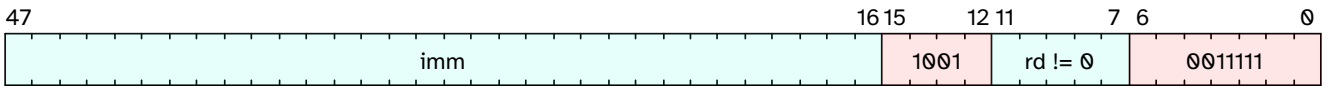
Synopsis

Or immediate

Mnemonic

```
qc.e.orai rd, imm
```

Encoding



Description

Or a 32-bit immediate `imm` to the value in `rd`, and store the result back in `rd`. Instruction encoded in QC.EAI instruction format.

Decode Variables

```
Bits<32> imm = $encoding[47:16];
Bits<5> rd = $encoding[11:7];
```

Operation

```
X[rd] = X[rd] | imm;
```

Included in

Extension	Version
Xqci	~> 0.1.0
Xqcilia	~> 0.1.0

6.58. qc.e.ori

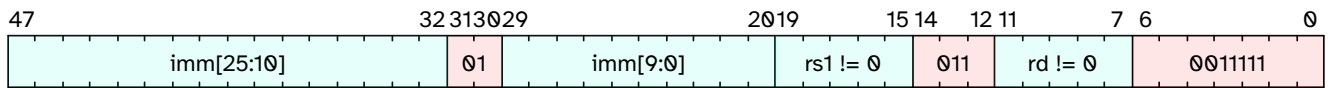
Synopsis

Or immediate

Mnemonic

```
qc.e.ori rd, rs1, imm
```

Encoding



Description

Or a sign-extended 26-bit immediate `imm` to the value in `rs1`, and store the result in `rd`. Instruction encoded in QC.EI instruction format.

Decode Variables

```
Bits<26> imm = {\$encoding\[47:32\], \$encoding\[29:20\]};  
Bits<5> rs1 = \$encoding\[19:15\];  
Bits<5> rd = \$encoding\[11:7\];
```

Operation

```
X[rd] = X[rs1] | sext(imm, 26);
```

Included in

Extension	Version
Xqci	~> 0.1.0
Xqcilia	~> 0.1.0

6.59. qc.e.sb

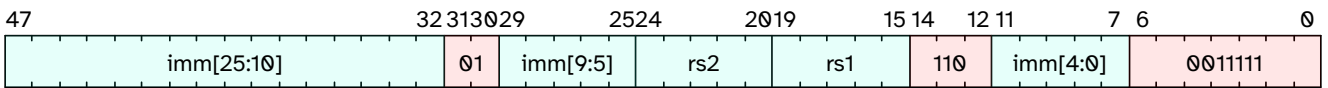
Synopsis

Store byte

Mnemonic

```
qc.e.sb  rs2, imm(rs1)
```

Encoding



Description

Store 8 bits of data from register `rs2` to an address formed by adding `rs1` to a signed offset `imm`. Instruction encoded in QC.ES instruction format.

Decode Variables

```
Bits<26> imm = { $encoding[47:32], $encoding[29:25], $encoding[11:7] };
Bits<5>  rs2 = $encoding[24:20];
Bits<5>  rs1 = $encoding[19:15];
```

Operation

```
XReg virtual_address = X[rs1] + $signed(imm);
write_memory<8>(virtual_address, X[rs2][7:0], $encoding);
```

Included in

Extension	Version
Xqci	~> 0.1.0
Xqcilo	~> 0.1.0

6.60. qc.e.sh

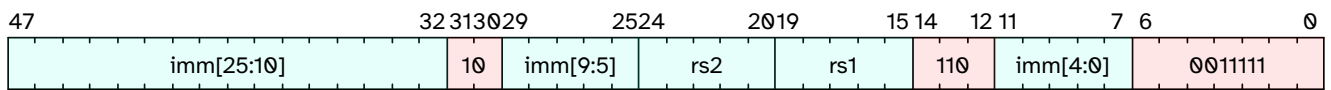
Synopsis

Store halfword

Mnemonic

```
qc.e.sh  rs2, imm(rs1)
```

Encoding



Description

Store 16 bits of data from register `rs2` to an address formed by adding `rs1` to a signed offset `imm`.
Instruction encoded in QC.ES instruction format.

Decode Variables

```
Bits<26> imm = { $encoding[47:32], $encoding[29:25], $encoding[11:7] };
Bits<5>  rs2 = $encoding[24:20];
Bits<5>  rs1 = $encoding[19:15];
```

Operation

```
XReg virtual_address = X[rs1] + $signed(imm);
write_memory<16>(virtual_address, X[rs2][15:0], $encoding);
```

Included in

Extension	Version
Xqci	~> 0.1.0
Xqcilo	~> 0.1.0

6.61. qc.e.sw

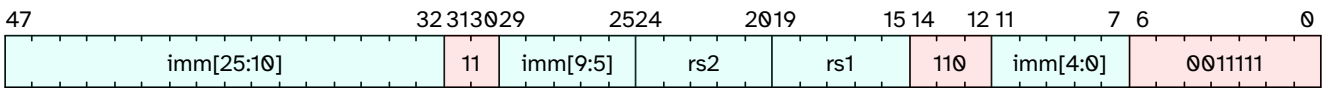
Synopsis

Store word

Mnemonic

```
qc.e.sw  rs2, imm(rs1)
```

Encoding



Description

Store 32 bits of data from register `rs2` to an address formed by adding `rs1` to a signed offset `imm`. Instruction encoded in QC.ES instruction format.

Decode Variables

```
Bits<26> imm = {\$encoding\[47:32\], \$encoding\[29:25\], \$encoding\[11:7\]};
Bits<5>  rs2 = \$encoding\[24:20\];
Bits<5>  rs1 = \$encoding\[19:15\];
```

Operation

```
XReg virtual_address = X[rs1] + \$signed(imm);
write_memory<32>(virtual_address, X[rs2][31:0], \$encoding);
```

Included in

Extension	Version
Xqci	~> 0.1.0
Xqcilo	~> 0.1.0

6.62. qc.e.xorai

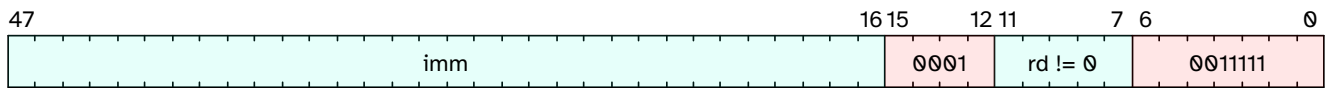
Synopsis

Exclusive Or immediate

Mnemonic

```
qc.e.xorai rd, imm
```

Encoding



Description

Exclusive or a 32-bit immediate `imm` to the value in `rd`, and store the result back in `rd`. Instruction encoded in QC.EAI instruction format.

Decode Variables

```
Bits<32> imm = $encoding[47:16];
Bits<5> rd = $encoding[11:7];
```

Operation

```
X[rd] = X[rd] ^ imm;
```

Included in

Extension	Version
Xqci	~> 0.1.0
Xqcilia	~> 0.1.0

6.63. qc.e.xori

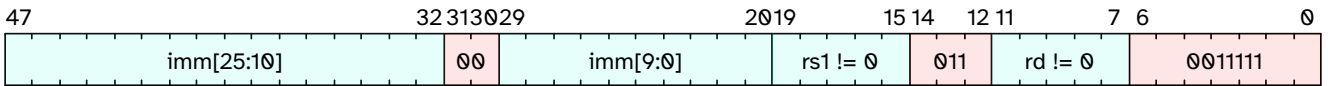
Synopsis

Exclusive Or immediate

Mnemonic

```
qc.e.xori rd, rs1, imm
```

Encoding



Description

Exclusive or a sign-extended 26-bit immediate `imm` to the value in `rs1`, and store the result in `rd`. Instruction encoded in QC.EI instruction format.

Decode Variables

```
Bits<26> imm = {\$encoding\[47:32\], \$encoding\[29:20\]};  
Bits<5> rs1 = \$encoding\[19:15\];  
Bits<5> rd = \$encoding\[11:7\];
```

Operation

```
X[rd] = X[rs1] ^ sext(imm, 26);
```

Included in

Extension	Version
Xqci	~> 0.1.0
Xqcilia	~> 0.1.0

6.64. qc.expand2

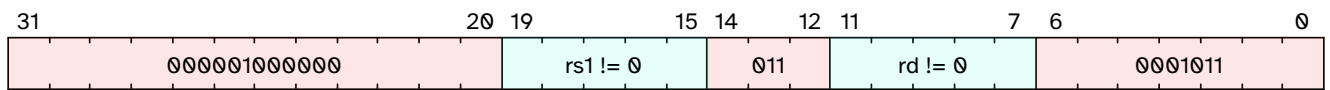
Synopsis

Bit expansion (every 2nd bit)

Mnemonic

```
qc.expand2  rd, rs1
```

Encoding



Description

Bit expansion (every 2nd bit) of `rs1`, bits `[31:16]` of `rs1` are ignored. Write result to `rd`. Instruction encoded in I instruction format.

Decode Variables

```
Bits<5> rs1 = $encoding[19:15];
Bits<5> rd = $encoding[11:7];
```

Operation

```
XReg b0 = {X[rs1][3], X[rs1][3], X[rs1][2], X[rs1][2], X[rs1][1], X[rs1][1], X[rs1][0], X[rs1][0]};
XReg b1 = {X[rs1][7], X[rs1][7], X[rs1][6], X[rs1][6], X[rs1][5], X[rs1][5], X[rs1][4], X[rs1][4]};
XReg b2 = {X[rs1][11], X[rs1][11], X[rs1][10], X[rs1][10], X[rs1][9], X[rs1][9], X[rs1][8], X[rs1][8]};
XReg b3 = {X[rs1][15], X[rs1][15], X[rs1][14], X[rs1][14], X[rs1][13], X[rs1][13], X[rs1][12], X[rs1][12]};
X[rd] = {b3[7:0], b2[7:0], b1[7:0], b0[7:0]};
```

Included in

Extension	Version
Xqci	~> 0.1.0
Xqcibm	~> 0.1.0

6.65. qc.expand3

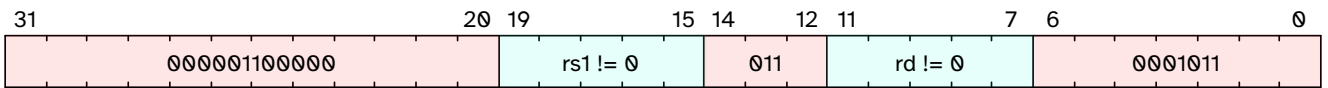
Synopsis

Bit expansion (every 3rd bit)

Mnemonic

```
qc.expand3  rd, rs1
```

Encoding



Description

Bit expansion (every 3rd bit) of rs1, bits [31:11] of rs1 are ignored. Write result to rd. Instruction encoded in I instruction format.

Decode Variables

```
Bits<5> rs1 = $encoding[19:15];
Bits<5> rd = $encoding[11:7];
```

Operation

```
XReg b0 = {X[rs1][2], X[rs1][2], X[rs1][1], X[rs1][1], X[rs1][1], X[rs1][0], X
[rs1][0], X[rs1][0]};
XReg b1 = {X[rs1][5], X[rs1][4], X[rs1][4], X[rs1][4], X[rs1][3], X[rs1][3], X
[rs1][3], X[rs1][2]};
XReg b2 = {X[rs1][7], X[rs1][7], X[rs1][7], X[rs1][6], X[rs1][6], X[rs1][6], X
[rs1][5], X[rs1][5]};
XReg b3 = {X[rs1][10], X[rs1][10], X[rs1][9], X[rs1][9], X[rs1][9], X[rs1][8],
X[rs1][8], X[rs1][8]};
X[rd] = {b3[7:0], b2[7:0], b1[7:0], b0[7:0]};
```

Included in

Extension	Version
Xqci	~> 0.1.0
Xqcibm	~> 0.1.0

6.66. qc.ext

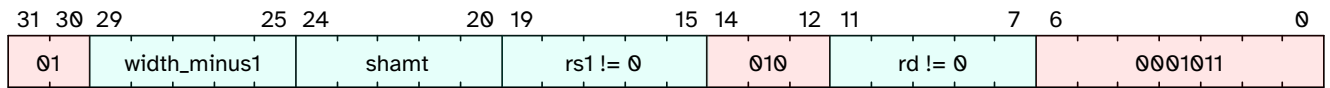
Synopsis

Extract bits signed

Mnemonic

```
qc.ext rd, rs1, width, shamt
```

Encoding



Description

Extract a subset of bits from `rs1` into `rd`, and sign-extend the result. The width of the subset is determined by $(width_minus1 + 1)$ (1..32), and the offset of the subset is determined by `shamt`. Instruction encoded in I instruction format.

Decode Variables

```
Bits<5> width_minus1 = $encoding[29:25];
Bits<5> shamt = $encoding[24:20];
Bits<5> rs1 = $encoding[19:15];
Bits<5> rd = $encoding[11:7];
```

Operation

```
XReg width = width_minus1 + 1;
XReg unsigned_extraction = (X[rs1] >> shamt) & ((1 << width) - 1);
X[rd] = sext(unsigned_extraction, width);
```

Included in

Extension	Version
Xqci	~> 0.1.0
Xqcibm	~> 0.1.0

6.67. qc.extd

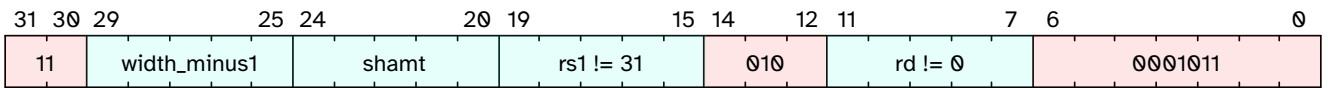
Synopsis

Extract bits from pair signed (Immediate)

Mnemonic

```
qc.extd rd, rs1, width, shamt
```

Encoding



Description

Extract a subset of bits from the register pair [rs1, rs1+1] into rd, and sign extend the result. The width of the subset is determined by (width_minus1 + 1) (1..32), and the offset (into the pair) of the subset is determined by shamt. Instruction encoded in I instruction format.

Decode Variables

```
Bits<5> width_minus1 = $encoding[29:25];
Bits<5> shamt = $encoding[24:20];
Bits<5> rs1 = $encoding[19:15];
Bits<5> rd = $encoding[11:7];
```

Operation

```
Bits<{1'b0, MXLEN} * 2> pair = {X[rs1 + 1], X[rs1]};
XReg width = width_minus1 + 1;
X[rd] = sext((pair >> shamt) & ((1 << width) - 1), width);
```

Included in

Extension	Version
Xqci	~> 0.1.0
Xqcibm	~> 0.1.0

6.68. qc.extdpr

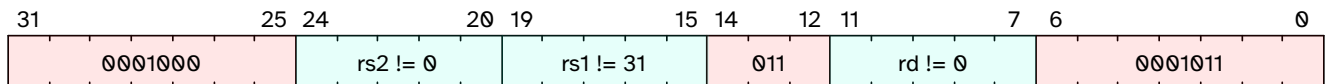
Synopsis

Extract bits from pair signed, packed descriptor (Register)

Mnemonic

```
qc.extdpr rd, rs1, rs2
```

Encoding



Description

Extract a subset of bits from the register pair [rs1, rs1+1] into rd, and sign extend the result. The width of the subset is determined by rs2 bits [13:8] (0..32), and the offset of the subset is determined by rs2 bits [5:0] (0..63). In case when rs2 bit [13] == 1 width is enforced to 32. In case when width == 0, to the destination register written 0. Instruction encoded in R instruction format.

Decode Variables

```
Bits<5> rs2 = $encoding[24:20];
Bits<5> rs1 = $encoding[19:15];
Bits<5> rd = $encoding[11:7];
```

Operation

```
Bits<{1'b0, MXLEN} * 2> pair = {X[rs1 + 1], X[rs1]};
XReg width_bits = X[rs2][13:8];
XReg width = (width_bits > 32) ? 32 : width_bits;
XReg shamt = X[rs2][5:0];
if (width > 0) {
    X[rd] = sext((pair >> shamt) & ((1 << width) - 1), width);
} else {
    X[rd] = 0;
}
```

Included in

Extension	Version
Xqci	~> 0.1.0
Xqcibm	~> 0.1.0

6.69. qc.extdprh

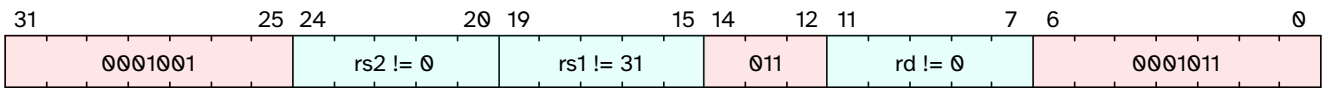
Synopsis

Extract bits from pair signed, packed descriptor high part (Register)

Mnemonic

```
qc.extdprh rd, rs1, rs2
```

Encoding



Description

Extract a subset of bits from the register pair [rs1, rs1+1] into rd, and sign extend the result. The width of the subset is determined by rs2 bits [29:24] (0..32), and the offset of the subset is determined by rs2 bits [21:16] (0..63). In case when rs2 bit [29] == 1 width is enforced to 32. In case when width == 0, to the destination register written 0. Instruction encoded in R instruction format.

Decode Variables

```
Bits<5> rs2 = $encoding[24:20];
Bits<5> rs1 = $encoding[19:15];
Bits<5> rd = $encoding[11:7];
```

Operation

```
Bits<{1'b0, MXLEN} * 2> pair = {X[rs1 + 1], X[rs1]};
XReg width_bits = X[rs2][29:24];
XReg width = (width_bits > 32) ? 32 : width_bits;
XReg shamt = X[rs2][21:16];
if (width > 0) {
    X[rd] = sext((pair >> shamt) & ((1 << width) - 1), width);
} else {
    X[rd] = 0;
}
```

Included in

Extension	Version
Xqci	~> 0.1.0
Xqcibm	~> 0.1.0

6.70. qc.extdr

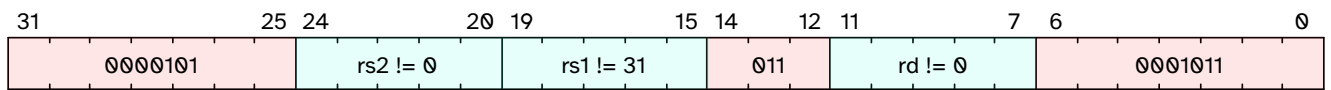
Synopsis

Extract bits from pair signed (Register)

Mnemonic

```
qc.extdr rd, rs1, rs2
```

Encoding



Description

Extract a subset of bits from the register pair [rs1, rs1+1] into rd, and sign extend the result. The width of the subset is determined by rs2 bits [21:16] (0..32), and the offset of the subset is determined by rs2 bits [5:0] (0..63). In case when rs2 bit [21] == 1 width is enforced to 32. In case when width == 0, to the destination register written 0. Instruction encoded in R instruction format.

Decode Variables

```
Bits<5> rs2 = $encoding[24:20];
Bits<5> rs1 = $encoding[19:15];
Bits<5> rd = $encoding[11:7];
```

Operation

```
Bits<{1'b0, MXLEN} * 2> pair = {X[rs1 + 1], X[rs1]};
XReg width_bits = X[rs2][21:16];
XReg width = (width_bits > 32) ? 32 : width_bits;
XReg shamt = X[rs2][5:0];
if (width > 0) {
    X[rd] = sext((pair >> shamt) & ((1 << width) - 1), width);
} else {
    X[rd] = 0;
}
```

Included in

Extension	Version
Xqci	~> 0.1.0
Xqcibm	~> 0.1.0

6.71. qc.extdu

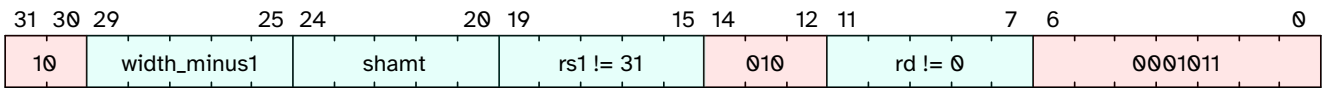
Synopsis

Extract bits from pair unsigned (Immediate)

Mnemonic

```
qc.extdu rd, rs1, width, shamt
```

Encoding



Description

Extract a subset of bits from the register pair [rs1, rs1+1] into rd. The width of the subset is determined by (width_minus1 + 1) (1..32), and the offset (into the pair) of the subset is determined by shamt. Instruction encoded in I instruction format.

Decode Variables

```
Bits<5> width_minus1 = $encoding[29:25];
Bits<5> shamt = $encoding[24:20];
Bits<5> rs1 = $encoding[19:15];
Bits<5> rd = $encoding[11:7];
```

Operation

```
Bits<{1'b0, MXLEN} * 2> pair = {X[rs1 + 1], X[rs1]};
XReg width = width_minus1 + 1;
X[rd] = (pair >> shamt) & ((1 << width) - 1);
```

Included in

Extension	Version
Xqci	~> 0.1.0
Xqcibm	~> 0.1.0

6.72. qc.extdupr

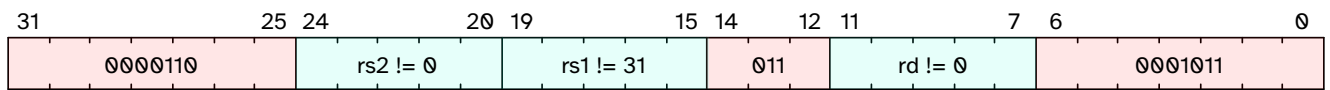
Synopsis

Extract bits from pair unsigned, packed descriptor (Register)

Mnemonic

```
qc.extdupr rd, rs1, rs2
```

Encoding



Description

Extract a subset of bits from the register pair [rs1, rs1+1] into rd. The width of the subset is determined by rs2 bits [13:8] (0..32), and the offset of the subset is determined by rs2 bits [5:0] (0..63). In case when rs2 bit [13] == 1 width is enforced to 32. In case when width == 0, to the destination register written 0. Instruction encoded in R instruction format.

Decode Variables

```
Bits<5> rs2 = $encoding[24:20];
Bits<5> rs1 = $encoding[19:15];
Bits<5> rd = $encoding[11:7];
```

Operation

```
Bits<{1'b0, MXLEN} * 2> pair = {X[rs1 + 1], X[rs1]};
XReg width_bits = X[rs2][13:8];
XReg width = (width_bits > 32) ? 32 : width_bits;
XReg shamt = X[rs2][5:0];
if (width > 0) {
    X[rd] = (pair >> shamt) & ((1 << width) - 1);
} else {
    X[rd] = 0;
}
```

Included in

Extension	Version
Xqci	~> 0.1.0
Xqcibm	~> 0.1.0

6.73. qc.extduprh

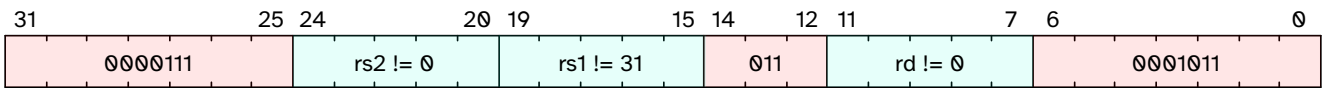
Synopsis

Extract bits from pair unsigned, packed descriptor high part (Register)

Mnemonic

```
qc.extduprh rd, rs1, rs2
```

Encoding



Description

Extract a subset of bits from the register pair [rs1, rs1+1] into rd. The width of the subset is determined by rs2 bits [29:24] (0..32), and the offset of the subset is determined by rs2 bits [21:16] (0..63). In case when rs2 bit [29] == 1 width is enforced to 32. In case when width == 0, to the destination register written 0. Instruction encoded in R instruction format.

Decode Variables

```
Bits<5> rs2 = $encoding[24:20];
Bits<5> rs1 = $encoding[19:15];
Bits<5> rd = $encoding[11:7];
```

Operation

```
Bits<{1'b0, MXLEN} * 2> pair = {X[rs1 + 1], X[rs1]};
XReg width_bits = X[rs2][29:24];
XReg width = (width_bits > 32) ? 32 : width_bits;
XReg shamt = X[rs2][21:16];
if (width > 0) {
    X[rd] = (pair >> shamt) & ((1 << width) - 1);
} else {
    X[rd] = 0;
}
```

Included in

Extension	Version
Xqci	~> 0.1.0
Xqcibm	~> 0.1.0

6.74. qc.extdur

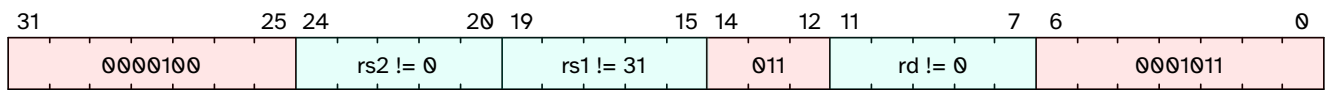
Synopsis

Extract bits from pair unsigned (Register)

Mnemonic

```
qc.extdur rd, rs1, rs2
```

Encoding



Description

Extract a subset of bits from the register pair [rs1, rs1+1] into rd. The width of the subset is determined by rs2 bits [21:16] (0..32), and the offset of the subset is determined by rs2 bits [5:0] (0..63). In case when rs2 bit [21] == 1 width is enforced to 32. In case when width == 0, to the destination register written 0. Instruction encoded in R instruction format.

Decode Variables

```
Bits<5> rs2 = $encoding[24:20];
Bits<5> rs1 = $encoding[19:15];
Bits<5> rd = $encoding[11:7];
```

Operation

```
Bits<{1'b0, MXLEN} * 2> pair = {X[rs1 + 1], X[rs1]};
XReg width_bits = X[rs2][21:16];
XReg width = (width_bits > 32) ? 32 : width_bits;
XReg shamt = X[rs2][5:0];
if (width > 0) {
    X[rd] = (pair >> shamt) & ((1 << width) - 1);
} else {
    X[rd] = 0;
}
```

Included in

Extension	Version
Xqci	~> 0.1.0
Xqcibm	~> 0.1.0

6.75. qc.extu

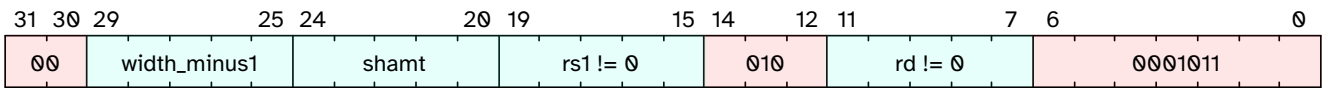
Synopsis

Extract bits unsigned

Mnemonic

```
qc.extu rd, rs1, width, shamt
```

Encoding



Description

Extract a subset of bits from `rs1` into `rd`. The width of the subset is determined by `(width_minus1 + 1)` (1..32), and the offset of the subset is determined by `shamt`. Instruction encoded in I instruction format.

Decode Variables

```
Bits<5> width_minus1 = $encoding[29:25];
Bits<5> shamt = $encoding[24:20];
Bits<5> rs1 = $encoding[19:15];
Bits<5> rd = $encoding[11:7];
```

Operation

```
XReg width = width_minus1 + 1;
X[rd] = (X[rs1] >> shamt) & ((1 << width) - 1);
```

Included in

Extension	Version
Xqci	~> 0.1.0
Xqcibm	~> 0.1.0

6.76. qc.insb

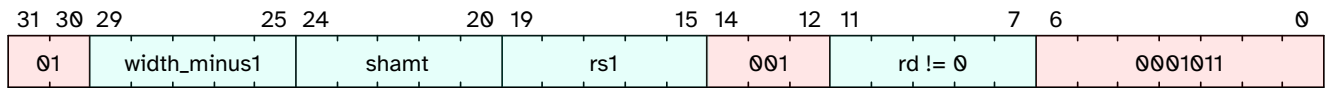
Synopsis

Insert bits (Register)

Mnemonic

```
qc.insb rd, rs1, width, shamt
```

Encoding



Description

Insertion of a subset of bits from `rs1` into `rd`. The width of the subset is determined by `(width_minus1 + 1)` (1..32), and the offset of the subset is determined by `shamt`. Instruction encoded in I instruction format.

Decode Variables

```
Bits<5> width_minus1 = $encoding[29:25];
Bits<5> shamt = $encoding[24:20];
Bits<5> rs1 = $encoding[19:15];
Bits<5> rd = $encoding[11:7];
```

Operation

```
XReg width = width_minus1 + 1;
XReg mask = ((1 << width) - 1) << shamt;
XReg orig_val = X[rd];
X[rd] = (orig_val & ~mask) | ((X[rs1] << shamt) & mask);
```

Included in

Extension	Version
Xqci	~> 0.1.0
Xqcibm	~> 0.1.0

6.77. qc.insbh

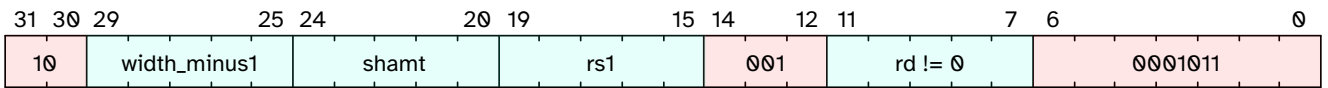
Synopsis

Insert bits in 64-bit higher part (Register)

Mnemonic

```
qc.insbh rd, rs1, width, shamt
```

Encoding



Description

Insertion of a subset of bits from `rs1` into `rd`. Instruction intended for insertion bits into bitfield within 64-bits, when bitfield crosses 32-bit boundary. Lower part of 64-bit destination is inserted using QC32.INSB, higher part using QC32.INSBH. The width of the subset is determined by $(width_minus1 + 1)$ (1..32), and the offset of the subset is determined by `shamt`. In case when $width + offset < 33$, the destination register is left unchanged. Instruction encoded in I instruction format.

Decode Variables

```
Bits<5> width_minus1 = $encoding[29:25];
Bits<5> shamt = $encoding[24:20];
Bits<5> rs1 = $encoding[19:15];
Bits<5> rd = $encoding[11:7];
```

Operation

```
XReg width = width_minus1 + 1;
if (width + shamt > 32) {
    XReg shifted_one = 1 << (width + shamt - 32);
    XReg mask = (shifted_one - 1);
    XReg orig_val = X[rd];
    XReg shifted_rs1 = X[rs1] >> (32 - shamt);
    X[rd] = (orig_val & ~mask) | (shifted_rs1 & mask);
}
```

Included in

Extension	Version
Xqci	~> 0.1.0
Xqcibm	~> 0.1.0

6.78. qc.insbhr

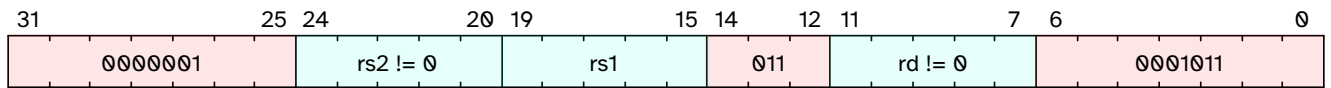
Synopsis

Insert bits in 64-bit higher part (Register)

Mnemonic

```
qc.insbhr rd, rs1, rs2
```

Encoding



Description

Insertion of a subset of bits from `rs1` into `rd`. Instruction intended for insertion bits into bitfield within 64-bits, when bitfield crosses 32-bit boundary. Lower part of 64-bit destination is inserted using QC32.INSBR, higher part using QC32.INSBHR. The width of the subset is determined by `rs2` bits [21:16] (0..32), and the offset of the subset is determined by `rs2` bits [4:0]. In case when width + offset < 33 or width == 0, the destination register is left unchanged. In case when `rs2` bit [21] == 1 width is enforced to 32. Instruction encoded in R instruction format.

Decode Variables

```
Bits<5> rs2 = $encoding[24:20];
Bits<5> rs1 = $encoding[19:15];
Bits<5> rd = $encoding[11:7];
```

Operation

```
XReg width_bits = X[rs2][21:16];
XReg width = (width_bits > 32) ? 32 : width_bits;
XReg shamt = X[rs2][4:0];
if ((width + shamt > 32) && (width > 0)) {
    XReg shifted_one = 1 << (width + shamt - 32);
    XReg mask = (shifted_one - 1);
    XReg orig_val = X[rd];
    XReg shifted_rs1 = X[rs1] >> (32 - shamt);
    X[rd] = (orig_val & ~mask) | (shifted_rs1 & mask);
}
```

Included in

Extension	Version
Xqci	~> 0.1.0
Xqcibm	~> 0.1.0

6.79. qc.insbi

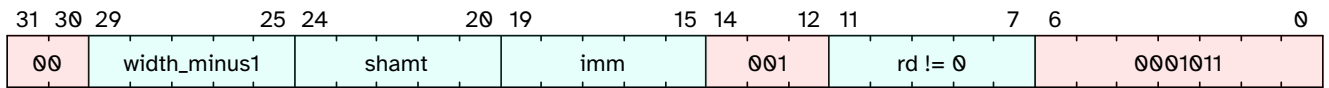
Synopsis

Insert bits (Immediate)

Mnemonic

```
qc.insbi rd, imm, width, shamt
```

Encoding



Description

Insertion of a subset of bits of an `imm` into `rd`. The width of the subset is determined by (`width_minus1` + 1) (1..32), and the offset of the subset is determined by `shamt`. Instruction encoded in I instruction format.

Decode Variables

```
Bits<5> width_minus1 = $encoding[29:25];
Bits<5> shamt = $encoding[24:20];
Bits<5> imm = $encoding[19:15];
Bits<5> rd = $encoding[11:7];
```

Operation

```
XReg width = width_minus1 + 1;
XReg mask = ((1 << width) - 1) << shamt;
XReg orig_val = X[rd];
X[rd] = (orig_val & ~mask) | ((imm << shamt) & mask);
```

Included in

Extension	Version
Xqci	~> 0.1.0
Xqcibm	~> 0.1.0

6.80. qc.insbpr

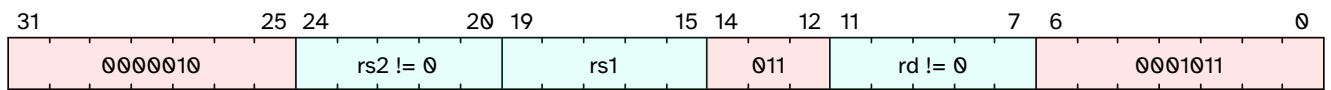
Synopsis

Insert bits, packed descriptor (Register)

Mnemonic

```
qc.insbpr rd, rs1, rs2
```

Encoding



Description

Insertion of a subset of bits from `rs1` into `rd`. The width of the subset is determined by `rs2` bits [13:8] (`0..32`), and the offset of the subset is determined by `rs2` bits [4:0]. In case when `width == 0`, the destination register is left unchanged. In case when `rs2` bit [13] == 1 width is enforced to 32. Instruction encoded in R instruction format.

Decode Variables

```
Bits<5> rs2 = $encoding[24:20];
Bits<5> rs1 = $encoding[19:15];
Bits<5> rd = $encoding[11:7];
```

Operation

```
XReg width_bits = X[rs2][13:8];
XReg width = (width_bits > 32) ? 32 : width_bits;
XReg shamt = X[rs2][4:0];
if (width > 0) {
    XReg mask = ((1 << width) - 1) << shamt;
    XReg orig_val = X[rd];
    X[rd] = (orig_val & ~mask) | ((X[rs1] << shamt) & mask);
}
```

Included in

Extension	Version
Xqci	~> 0.1.0
Xqcibm	~> 0.1.0

6.81. qc.insbprh

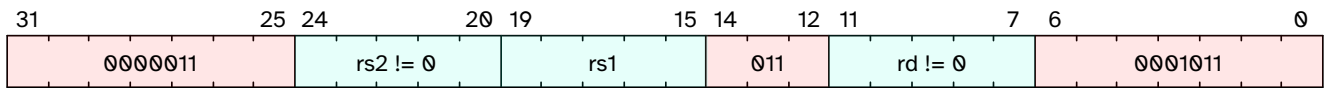
Synopsis

Insert bits, packed descriptor high part (Register)

Mnemonic

```
qc.insbprh rd, rs1, rs2
```

Encoding



Description

Insertion of a subset of bits from `rs1` into `rd`. The width of the subset is determined by `rs2` bits `[29:24]` (`0..32`), and the offset of the subset is determined by `rs2` bits `[20:16]`. In case when `width == 0`, the destination register is left unchanged. In case when `rs2` bit `[29] == 1` width is enforced to 32. Instruction encoded in R instruction format.

Decode Variables

```
Bits<5> rs2 = $encoding[24:20];
Bits<5> rs1 = $encoding[19:15];
Bits<5> rd = $encoding[11:7];
```

Operation

```
XReg width_bits = X[rs2][29:24];
XReg width = (width_bits > 32) ? 32 : width_bits;
XReg shamt = X[rs2][20:16];
if (width > 0) {
    XReg mask = ((1 << width) - 1) << shamt;
    XReg orig_val = X[rd];
    X[rd] = (orig_val & ~mask) | ((X[rs1] << shamt) & mask);
}
```

Included in

Extension	Version
Xqci	~> 0.1.0
Xqcibm	~> 0.1.0

6.82. qc.insbr

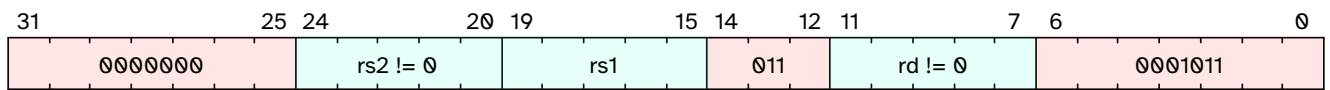
Synopsis

Insert bits (Register)

Mnemonic

```
qc.insbr rd, rs1, rs2
```

Encoding



Description

Insertion of a subset of bits from `rs1` into `rd`. The width of the subset is determined by `rs2` bits [21:16] (0..32), and the offset of the subset is determined by `rs2` bits [4:0]. In case when `width == 0`, the destination register is left unchanged. In case when `rs2` bit [21] == 1 width is enforced to 32. Instruction encoded in R instruction format.

Decode Variables

```
Bits<5> rs2 = $encoding[24:20];
Bits<5> rs1 = $encoding[19:15];
Bits<5> rd = $encoding[11:7];
```

Operation

```
XReg width_bits = X[rs2][21:16];
XReg width = (width_bits > 32) ? 32 : width_bits;
XReg shamt = X[rs2][4:0];
if (width > 0) {
    XReg mask = ((1 << width) - 1) << shamt;
    XReg orig_val = X[rd];
    X[rd] = (orig_val & ~mask) | ((X[rs1] << shamt) & mask);
}
```

Included in

Extension	Version
Xqci	~> 0.1.0
Xqcibm	~> 0.1.0

6.83. qc.insbri

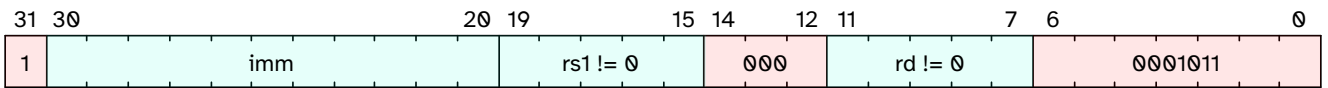
Synopsis

Insert bits (Immediate)

Mnemonic

```
qc.insbri rd, rs1, imm
```

Encoding



Description

Insertion of a subset of bits of an `imm` into `rd`. The width of the subset is determined by `rs1` bits [21:16] (0..32), and the offset of the subset is determined by `rs1` bits [4:0]. In case when `width == 0`, the destination register is left unchanged. In case when `rs1` bit [21] == 1 width is enforced to 32. Instruction encoded in I instruction format.

Decode Variables

```
Bits<11> imm = $encoding[30:20];
Bits<5> rs1 = $encoding[19:15];
Bits<5> rd = $encoding[11:7];
```

Operation

```
XReg width_bits = X[rs1][21:16];
XReg width = (width_bits > 32) ? 32 : width_bits;
XReg shamt = X[rs1][4:0];
if (width > 0) {
    XReg mask = ((1 << width) - 1) << shamt;
    XReg orig_val = X[rd];
    X[rd] = (orig_val & ~mask) | ((imm << shamt) & mask);
}
```

Included in

Extension	Version
Xqci	~> 0.1.0
Xqcibm	~> 0.1.0

6.84. qc.inw

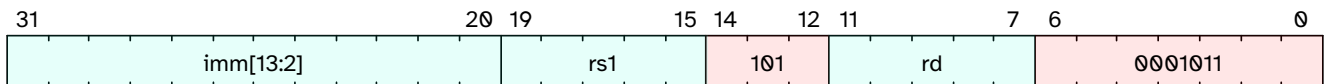
Synopsis

Input word from non-memory-mapped device

Mnemonic

```
qc.inw rd, imm(rs1)
```

Encoding



Description

Input 32 bits of data into register `rd` from a non-memory-mapped device. Such devices have own address space, unrelated to memory map. Device space address formed by adding `rs1` to to a unsigned offset `imm`. Instruction encoded in I instruction format.

Decode Variables

```
Bits<14> imm = {$encoding[31:20], 2'd0};
Bits<5> rs1 = $encoding[19:15];
Bits<5> rd = $encoding[11:7];
```

Operation

```
XReg device_address = X[rs1] + imm;
X[rd] = read_device_32(device_address);
```

Included in

Extension	Version
Xqci	~> 0.1.0
Xqciio	~> 0.1.0

6.85. qc.li

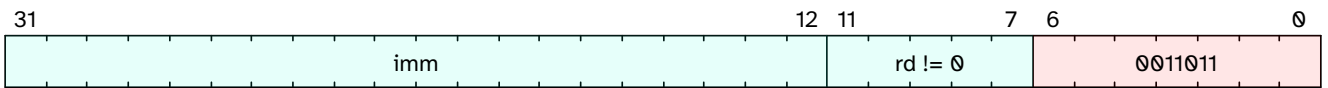
Synopsis

Load immediate large

Mnemonic

```
qc.li rd, imm
```

Encoding



Description

Loads the 20-bit immediate `imm` into `rd`. Instruction encoded in U instruction format.

Decode Variables

```
Bits<20> imm = { $encoding[31], $encoding[15:12], $encoding[30:16] };
Bits<5> rd = $encoding[11:7];
```

Operation

```
X[rd] = sext(imm, 20);
```

Included in

Extension	Version
Xqci	~> 0.1.0
Xqcili	~> 0.1.0

6.86. qc.lieq

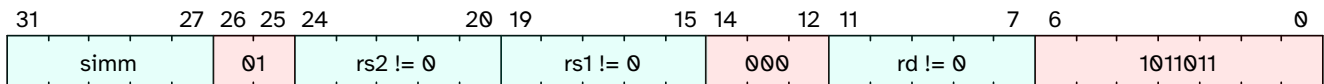
Synopsis

Conditional load immediate if equal (Register)

Mnemonic

```
qc.lieq rd, rs1, rs2, simm
```

Encoding



Description

Move `simm` to `rd` if the value in `rs1` is equal to value `rs2`. Instruction encoded in R instruction format.

Decode Variables

```
Bits<5> rs1 = $encoding[19:15];
Bits<5> rs2 = $encoding[24:20];
Bits<5> simm = $encoding[31:27];
Bits<5> rd = $encoding[11:7];
```

Operation

```
if (X[rs1] == X[rs2]) {
    X[rd] = sext(simm, 5);
}
```

Included in

Extension	Version
Xqci	~> 0.1.0
Xqcicli	~> 0.1.0

6.87. qc.lieqi

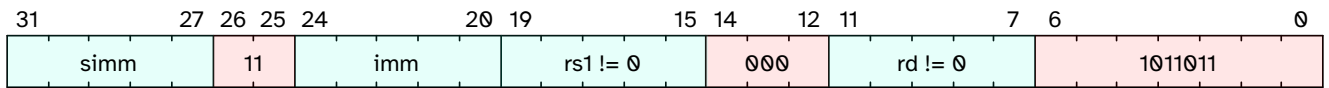
Synopsis

Conditional load immediate if equal (Immediate)

Mnemonic

```
qc.lieqi rd, rs1, imm, simm
```

Encoding



Description

Move `simm` to `rd` if the value in `rs1` is equal to value `imm`. Instruction encoded in I instruction format.

Decode Variables

```
Bits<5> rs1 = $encoding[19:15];
Bits<5> imm = $encoding[24:20];
Bits<5> simm = $encoding[31:27];
Bits<5> rd = $encoding[11:7];
```

Operation

```
if ($signed(X[rs1]) == $signed(imm)) {
    X[rd] = sext(simm, 5);
}
```

Included in

Extension	Version
Xqci	~> 0.1.0
Xqcicli	~> 0.1.0

6.88. qc.lige

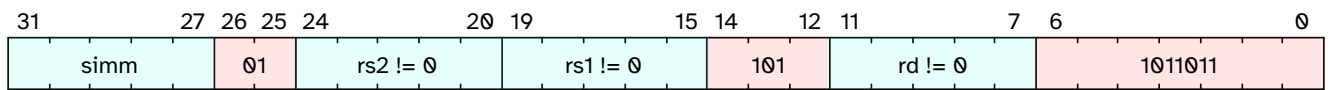
Synopsis

Conditional load immediate if great or equal than (Register)

Mnemonic

```
qc.lige rd, rs1, rs2, simm
```

Encoding



Description

Move `simm` to `rd` if the value in `rs1` is great or equal than value `rs2`. Instruction encoded in R instruction format.

Decode Variables

```
Bits<5> rs1 = $encoding[19:15];
Bits<5> rs2 = $encoding[24:20];
Bits<5> simm = $encoding[31:27];
Bits<5> rd = $encoding[11:7];
```

Operation

```
if ($signed(X[rs1]) >= $signed(X[rs2])) {
    X[rd] = sext(simm, 5);
}
```

Included in

Extension	Version
Xqci	~> 0.1.0
Xqcicli	~> 0.1.0

6.89. qc.ligei

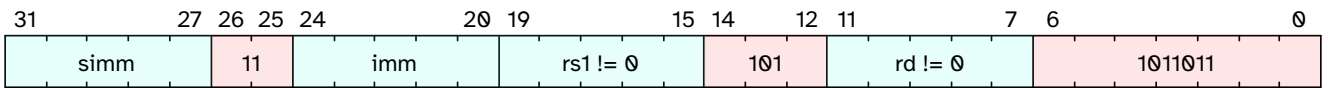
Synopsis

Conditional load immediate if great or equal than (Immediate)

Mnemonic

```
qc.ligei rd, rs1, imm, simm
```

Encoding



Description

Move `simm` to `rd` if the value in `rs1` is great or equal than value `imm`. Instruction encoded in I instruction format.

Decode Variables

```
Bits<5> rs1 = $encoding[19:15];
Bits<5> imm = $encoding[24:20];
Bits<5> simm = $encoding[31:27];
Bits<5> rd = $encoding[11:7];
```

Operation

```
if ($signed(X[rs1]) >= $signed(imm)) {
    X[rd] = sext(simm, 5);
}
```

Included in

Extension	Version
Xqci	~> 0.1.0
Xqcicli	~> 0.1.0

6.90. qc.ligeu

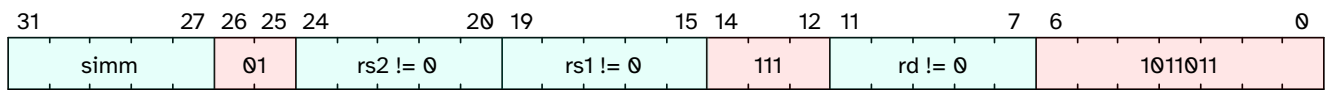
Synopsis

Conditional load immediate if great or equal than unsigned (Register)

Mnemonic

```
qc.ligeu rd, rs1, rs2, simm
```

Encoding



Description

Move `simm` to `rd` if the unsigned value in `rs1` is great or equal than unsigned value `rs2`. Instruction encoded in R instruction format.

Decode Variables

```
Bits<5> rs1 = $encoding[19:15];
Bits<5> rs2 = $encoding[24:20];
Bits<5> simm = $encoding[31:27];
Bits<5> rd = $encoding[11:7];
```

Operation

```
if (X[rs1] >= X[rs2]) {
    X[rd] = sext(simm, 5);
}
```

Included in

Extension	Version
Xqci	~> 0.1.0
Xqcicli	~> 0.1.0

6.91. qc.ligeui

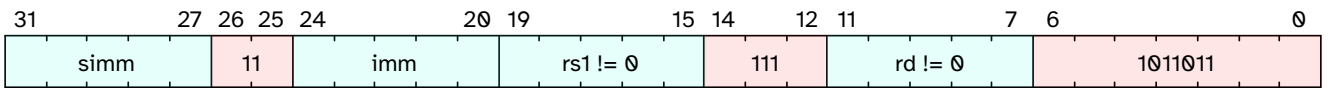
Synopsis

Conditional load immediate if great or equal than unsigned (Immediate)

Mnemonic

```
qc.ligeui rd, rs1, imm, simm
```

Encoding



Description

Move `simm` to `rd` if the unsigned value in `rs1` is great or equal than unsigned value `imm`. Instruction encoded in I instruction format.

Decode Variables

```
Bits<5> rs1 = $encoding[19:15];
Bits<5> imm = $encoding[24:20];
Bits<5> simm = $encoding[31:27];
Bits<5> rd = $encoding[11:7];
```

Operation

```
if (X[rs1] >= imm) {
    X[rd] = sext(simm, 5);
}
```

Included in

Extension	Version
Xqci	~> 0.1.0
Xqcicli	~> 0.1.0

6.92. qc.lilt

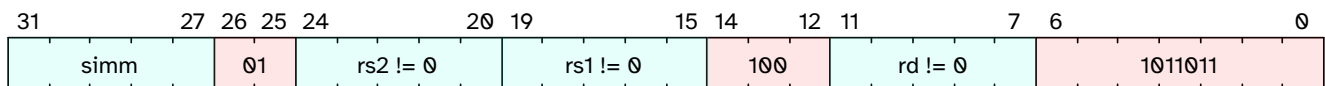
Synopsis

Conditional load immediate if less than (Register)

Mnemonic

```
qc.lilt rd, rs1, rs2, simm
```

Encoding



Description

Move `simm` to `rd` if the value in `rs1` is less than value `rs2`. Instruction encoded in R instruction format.

Decode Variables

```
Bits<5> rs1 = $encoding[19:15];
Bits<5> rs2 = $encoding[24:20];
Bits<5> simm = $encoding[31:27];
Bits<5> rd = $encoding[11:7];
```

Operation

```
if ($signed(X[rs1]) < $signed(X[rs2])) {
    X[rd] = sext(simm, 5);
}
```

Included in

Extension	Version
Xqci	~> 0.1.0
Xqcicli	~> 0.1.0

6.93. qc.lilti

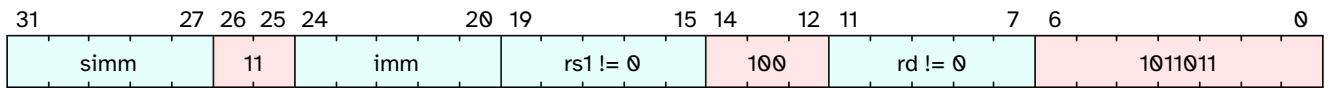
Synopsis

Conditional load immediate if less than (Immediate)

Mnemonic

```
qc.lilti rd, rs1, imm, simm
```

Encoding



Description

Move `simm` to `rd` if the value in `rs1` is less than value `imm`. Instruction encoded in I instruction format.

Decode Variables

```
Bits<5> rs1 = $encoding[19:15];
Bits<5> imm = $encoding[24:20];
Bits<5> simm = $encoding[31:27];
Bits<5> rd = $encoding[11:7];
```

Operation

```
if ($signed(X[rs1]) < $signed(imm)) {
    X[rd] = sext(simm, 5);
}
```

Included in

Extension	Version
Xqci	~> 0.1.0
Xqcicli	~> 0.1.0

6.94. qc.liltu

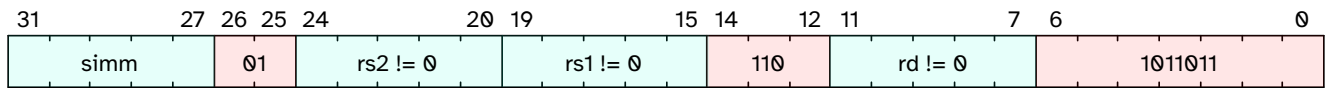
Synopsis

Conditional load immediate if less than unsigned (Register)

Mnemonic

```
qc.liltu rd, rs1, rs2, simm
```

Encoding



Description

Move `simm` to `rd` if the unsigned value in `rs1` is less than unsigned value `rs2`. Instruction encoded in R instruction format.

Decode Variables

```
Bits<5> rs1 = $encoding[19:15];
Bits<5> rs2 = $encoding[24:20];
Bits<5> simm = $encoding[31:27];
Bits<5> rd = $encoding[11:7];
```

Operation

```
if (X[rs1] < X[rs2]) {
    X[rd] = sext(simm, 5);
}
```

Included in

Extension	Version
Xqci	~> 0.1.0
Xqcicli	~> 0.1.0

6.95. qc.liltui

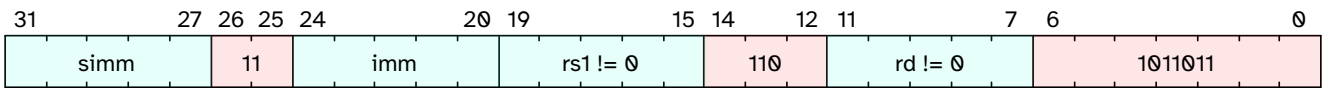
Synopsis

Conditional load immediate if less than unsigned (Immediate)

Mnemonic

```
qc.liltui rd, rs1, imm, simm
```

Encoding



Description

Move `simm` to `rd` if the unsigned value in `rs1` is less than unsigned value `imm`. Instruction encoded in I instruction format.

Decode Variables

```
Bits<5> rs1 = $encoding[19:15];
Bits<5> imm = $encoding[24:20];
Bits<5> simm = $encoding[31:27];
Bits<5> rd = $encoding[11:7];
```

Operation

```
if (X[rs1] < imm) {
    X[rd] = sext(simm, 5);
}
```

Included in

Extension	Version
Xqci	~> 0.1.0
Xqcicli	~> 0.1.0

6.96. qc.line

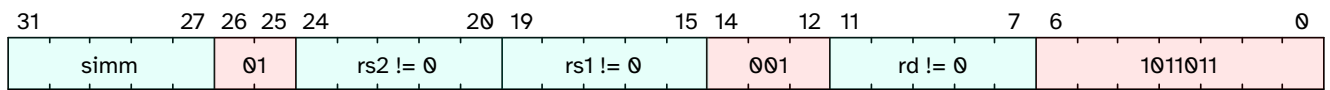
Synopsis

Conditional load immediate if not equal (Register)

Mnemonic

```
qc.line  rd, rs1, rs2, simm
```

Encoding



Description

Move `simm` to `rd` if the value in `rs1` is not equal to value `rs2`. Instruction encoded in R instruction format.

Decode Variables

```
Bits<5> rs1 = $encoding[19:15];
Bits<5> rs2 = $encoding[24:20];
Bits<5> simm = $encoding[31:27];
Bits<5> rd = $encoding[11:7];
```

Operation

```
if ($signed(X[rs1]) != $signed(X[rs2])) {
    X[rd] = sext(simm, 5);
}
```

Included in

Extension	Version
Xqci	~> 0.1.0
Xqcicli	~> 0.1.0

6.97. qc.linei

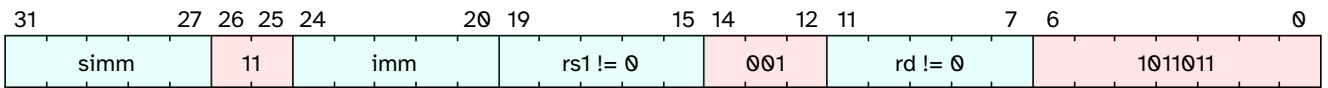
Synopsis

Conditional load immediate if not equal (Immediate)

Mnemonic

```
qc.linei rd, rs1, imm, simm
```

Encoding



Description

Move `simm` to `rd` if the value in `rs1` is not equal to value `imm`. Instruction encoded in I instruction format.

Decode Variables

```
Bits<5> rs1 = $encoding[19:15];
Bits<5> imm = $encoding[24:20];
Bits<5> simm = $encoding[31:27];
Bits<5> rd = $encoding[11:7];
```

Operation

```
if ($signed(X[rs1]) != $signed(imm)) {
    X[rd] = sext(simm, 5);
}
```

Included in

Extension	Version
Xqci	~> 0.1.0
Xqcicli	~> 0.1.0

6.98. qc.lrb

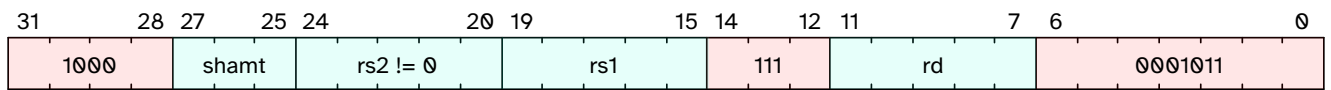
Synopsis

Load indexed byte

Mnemonic

```
qc.lrb rd, rs1, rs2, shamt
```

Encoding



Description

Load 8 bits of data into register `rd` from an address formed by adding `rs1` to `rs2`, shifted by `shamt`. Sign extend the result. Instruction encoded in R instruction format.

Decode Variables

```
Bits<3> shamt = $encoding[27:25];
Bits<5> rs1 = $encoding[19:15];
Bits<5> rs2 = $encoding[24:20];
Bits<5> rd = $encoding[11:7];
```

Operation

```
XReg virtual_address = X[rs1] + (X[rs2] << shamt);
X[rd] = sext(read_memory<8>(virtual_address, $encoding), 8);
```

Included in

Extension	Version
Xqci	~> 0.1.0
Xqcisls	~> 0.1.0

6.99. qc.lrbu

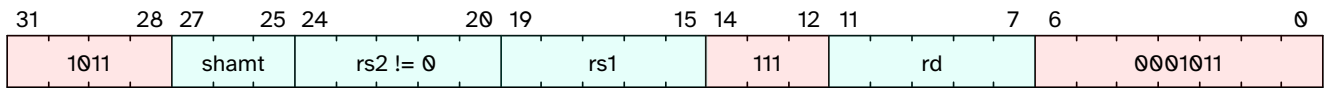
Synopsis

Load indexed unsigned byte

Mnemonic

```
qc.lrbu rd, rs1, rs2, shamt
```

Encoding



Description

Load 8 bits of data into register `rd` from an address formed by adding `rs1` to `rs2`, shifted by `shamt`. Instruction encoded in R instruction format.

Decode Variables

```
Bits<3> shamt = $encoding[27:25];
Bits<5> rs1 = $encoding[19:15];
Bits<5> rs2 = $encoding[24:20];
Bits<5> rd = $encoding[11:7];
```

Operation

```
XReg virtual_address = X[rs1] + (X[rs2] << shamt);
X[rd] = read_memory<8>(virtual_address, $encoding);
```

Included in

Extension	Version
Xqci	~> 0.1.0
Xqcisls	~> 0.1.0

6.100. qc.lrh

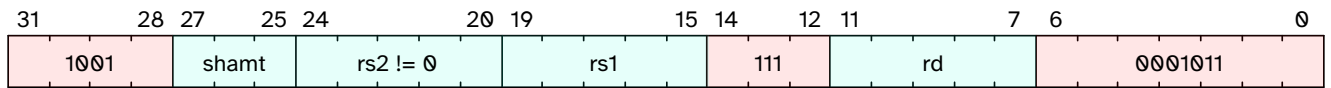
Synopsis

Load indexed halfword

Mnemonic

```
qc.lrh rd, rs1, rs2, shamt
```

Encoding



Description

Load 16 bits of data into register `rd` from an address formed by adding `rs1` to `rs2`, shifted by `shamt`. Sign extend the result. Instruction encoded in R instruction format.

Decode Variables

```
Bits<3> shamt = $encoding[27:25];
Bits<5> rs1 = $encoding[19:15];
Bits<5> rs2 = $encoding[24:20];
Bits<5> rd = $encoding[11:7];
```

Operation

```
XReg virtual_address = X[rs1] + (X[rs2] << shamt);
X[rd] = sext(read_memory<16>(virtual_address, $encoding), 16);
```

Included in

Extension	Version
Xqci	~> 0.1.0
Xqcisls	~> 0.1.0

6.101. qc.lrhu

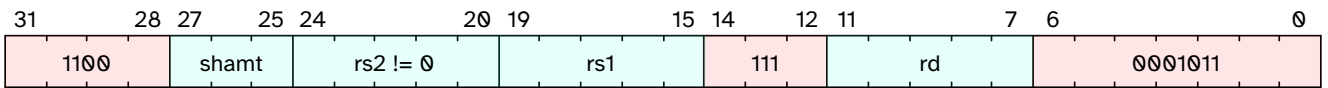
Synopsis

Load indexed unsigned halfword

Mnemonic

```
qc.lrhu rd, rs1, rs2, shamt
```

Encoding



Description

Load 16 bits of data into register `rd` from an address formed by adding `rs1` to `rs2`, shifted by `shamt`. Instruction encoded in R instruction format.

Decode Variables

```
Bits<3> shamt = $encoding[27:25];
Bits<5> rs1 = $encoding[19:15];
Bits<5> rs2 = $encoding[24:20];
Bits<5> rd = $encoding[11:7];
```

Operation

```
XReg virtual_address = X[rs1] + (X[rs2] << shamt);
X[rd] = read_memory<16>(virtual_address, $encoding);
```

Included in

Extension	Version
Xqci	~> 0.1.0
Xqcisls	~> 0.1.0

6.102. qc.lrw

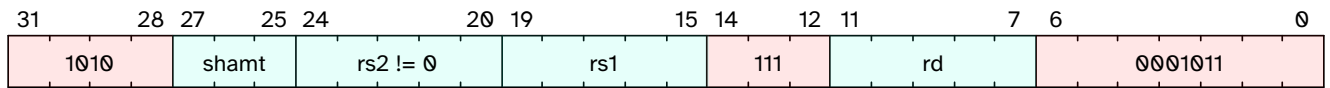
Synopsis

Load indexed word

Mnemonic

```
qc.lrw rd, rs1, rs2, shamt
```

Encoding



Description

Load 32 bits of data into register `rd` from an address formed by adding `rs1` to `rs2`, shifted by `shamt`. Instruction encoded in R instruction format.

Decode Variables

```
Bits<3> shamt = $encoding[27:25];
Bits<5> rs1 = $encoding[19:15];
Bits<5> rs2 = $encoding[24:20];
Bits<5> rd = $encoding[11:7];
```

Operation

```
XReg virtual_address = X[rs1] + (X[rs2] << shamt);
X[rd] = read_memory<32>(virtual_address, $encoding);
```

Included in

Extension	Version
Xqci	~> 0.1.0
Xqcisls	~> 0.1.0

6.103. qc.lwm

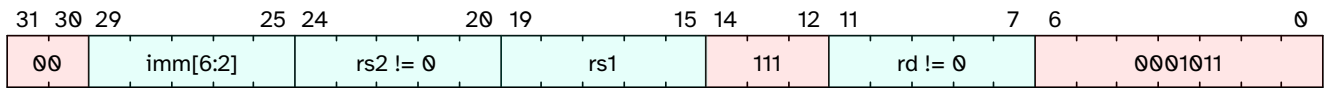
Synopsis

Load word multiple

Mnemonic

```
qc.lwm rd, rs2, imm(rs1)
```

Encoding



Description

Loads multiple words starting from address (rs1 + imm) to registers, starting from rd. The number of words is in rs2. Instruction encoded in R instruction format.

Decode Variables

```
Bits<7> imm = {$encoding[29:25], 2'd0};
Bits<5> rs1 = $encoding[19:15];
Bits<5> rs2 = $encoding[24:20];
Bits<5> rd = $encoding[11:7];
```

Operation

```
XReg vaddr = X[rs1] + imm;
XReg num_words = X[rs2];
raise(ExceptionCode::IllegalInstruction, effective_ldst_mode(), $encoding) if
(rd + num_words) > 32;
for (U32 i = 0; i < num_words; i++) {
    X[rd + i] = read_memory<32>(vaddr, $encoding);
    vaddr = vaddr + 4;
}
```

Included in

Extension	Version
Xqci	~> 0.1.0
Xqcilsm	~> 0.1.0

6.104. qc.lwmi

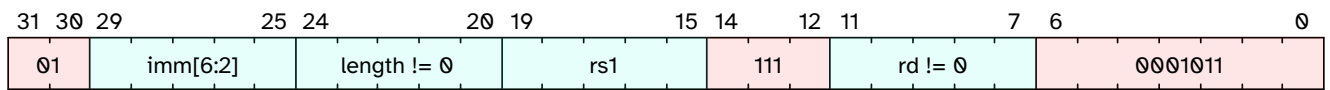
Synopsis

Load word multiple (Immediate)

Mnemonic

```
qc.lwmi rd, length, imm(rs1)
```

Encoding



Description

Loads multiple words starting from address (rs1 + imm) to registers, starting from rd. The number of words is in the length immediate. Instruction encoded in R instruction format.

Decode Variables

```
Bits<7> imm = {$encoding[29:25], 2'd0};
Bits<5> rs1 = $encoding[19:15];
Bits<5> length = $encoding[24:20];
Bits<5> rd = $encoding[11:7];
```

Operation

```
XReg vaddr = X[rs1] + imm;
raise(ExceptionCode::IllegalInstruction, effective_ldst_mode(), $encoding) if
(rd + length) > 32;
for (U32 i = 0; i < length; i++) {
    X[rd + i] = read_memory<32>(vaddr, $encoding);
    vaddr = vaddr + 4;
}
```

Included in

Extension	Version
Xqci	~> 0.1.0
Xqcilsm	~> 0.1.0

6.105. qc.muliadd

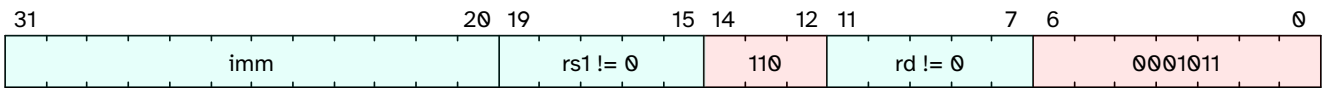
Synopsis

Multiply and accumulate (Immediate)

Mnemonic

```
qc.muliadd rd, rs1, imm
```

Encoding



Description

Increments `rd` by the multiplication of `rs1` and a signed immediate `imm`. Instruction encoded in I instruction format.

Decode Variables

```
Bits<12> imm = $encoding[31:20];
Bits<5> rs1 = $encoding[19:15];
Bits<5> rd = $encoding[11:7];
```

Operation

```
XReg orig_val = X[rd];
X[rd] = orig_val + (X[rs1] * sext(imm, 12));
```

Included in

Extension	Version
Xqci	~> 0.1.0
Xqciac	~> 0.1.0

6.106. qc.mveq

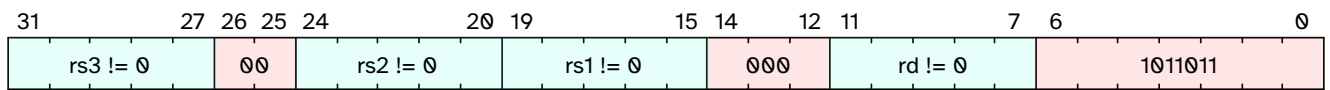
Synopsis

Conditional move if equal (Register)

Mnemonic

```
qc.mveq rd, rs1, rs2, rs3
```

Encoding



Description

Move rs3 to rd if the value in rs1 is equal to value rs2. Instruction encoded in R4 instruction format.

Decode Variables

```
Bits<5> rs1 = $encoding[19:15];
Bits<5> rs2 = $encoding[24:20];
Bits<5> rs3 = $encoding[31:27];
Bits<5> rd = $encoding[11:7];
```

Operation

```
if (X[rs1] == X[rs2]) {
    X[rd] = X[rs3];
}
```

Included in

Extension	Version
Xqci	~> 0.1.0
Xqcicm	~> 0.1.0

6.107. qc.mveqi

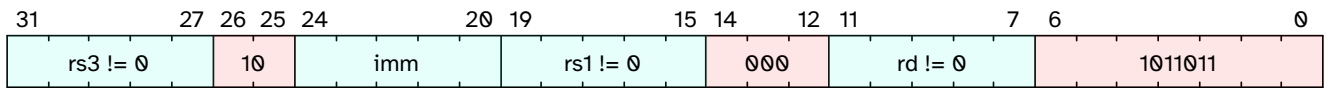
Synopsis

Conditional move if equal (Immediate)

Mnemonic

```
qc.mveqi rd, rs1, imm, rs3
```

Encoding



Description

Move `rs3` to `rd` if the value in `rs1` is equal to value of `imm`. Instruction encoded in R4 instruction format.

Decode Variables

```
Bits<5> rs1 = $encoding[19:15];
Bits<5> imm = $encoding[24:20];
Bits<5> rs3 = $encoding[31:27];
Bits<5> rd = $encoding[11:7];
```

Operation

```
if ($signed(X[rs1]) == $signed(imm)) {
    X[rd] = X[rs3];
}
```

Included in

Extension	Version
Xqci	~> 0.1.0
Xqcicm	~> 0.1.0

6.108. qc.mvge

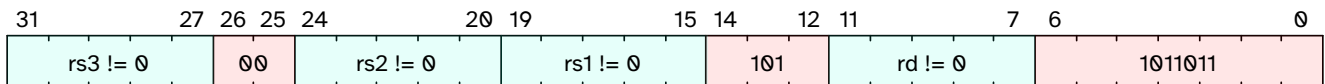
Synopsis

Conditional move if great or equal than (Register)

Mnemonic

```
qc.mvge rd, rs1, rs2, rs3
```

Encoding



Description

Move rs3 to rd if the value in rs1 is great or equal than value rs2. Instruction encoded in R4 instruction format.

Decode Variables

```
Bits<5> rs1 = $encoding[19:15];
Bits<5> rs2 = $encoding[24:20];
Bits<5> rs3 = $encoding[31:27];
Bits<5> rd = $encoding[11:7];
```

Operation

```
if ($signed(X[rs1]) >= $signed(X[rs2])) {
    X[rd] = X[rs3];
}
```

Included in

Extension	Version
Xqci	~> 0.1.0
Xqcicm	~> 0.1.0

6.109. qc.mvgei

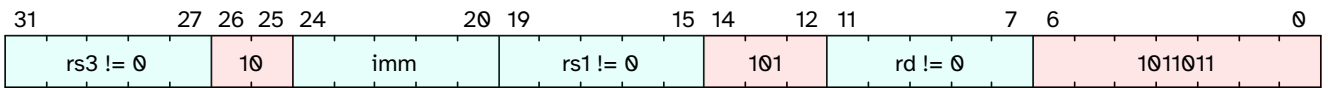
Synopsis

Conditional move if great or equal than (Immediate)

Mnemonic

```
qc.mvgei rd, rs1, imm, rs3
```

Encoding



Description

Move rs3 to rd if the value in rs1 is great or equal than value of imm. Instruction encoded in R4 instruction format.

Decode Variables

```
Bits<5> rs1 = $encoding[19:15];
Bits<5> imm = $encoding[24:20];
Bits<5> rs3 = $encoding[31:27];
Bits<5> rd = $encoding[11:7];
```

Operation

```
if ($signed(X[rs1]) >= $signed(imm)) {
    X[rd] = X[rs3];
}
```

Included in

Extension	Version
Xqci	~> 0.1.0
Xqcicm	~> 0.1.0

6.110. qc.mvgeu

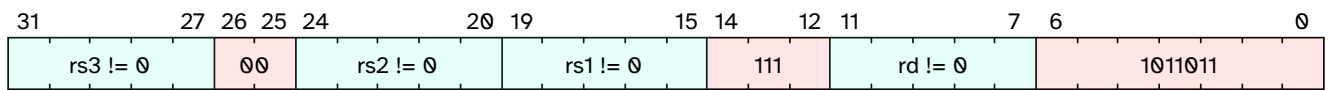
Synopsis

Conditional move if great or equal than unsigned (Register)

Mnemonic

```
qc.mvgeu rd, rs1, rs2, rs3
```

Encoding



Description

Move `rs3` to `rd` if the unsigned value in `rs1` is great or equal than unsigned value `rs2`. Instruction encoded in R4 instruction format.

Decode Variables

```
Bits<5> rs1 = $encoding[19:15];
Bits<5> rs2 = $encoding[24:20];
Bits<5> rs3 = $encoding[31:27];
Bits<5> rd = $encoding[11:7];
```

Operation

```
if (X[rs1] >= X[rs2]) {
    X[rd] = X[rs3];
}
```

Included in

Extension	Version
Xqci	~> 0.1.0
Xqcicm	~> 0.1.0

6.111. qc.mvgeui

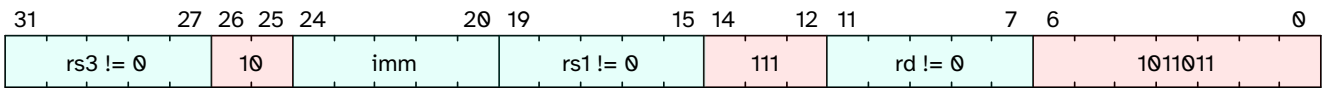
Synopsis

Conditional move if great or equal than unsigned (Immediate)

Mnemonic

```
qc.mvgeui rd, rs1, imm, rs3
```

Encoding



Description

Move `rs3` to `rd` if the unsigned value in `rs1` is great or equal than unsigned value of `imm`. Instruction encoded in R4 instruction format.

Decode Variables

```
Bits<5> rs1 = $encoding[19:15];
Bits<5> imm = $encoding[24:20];
Bits<5> rs3 = $encoding[31:27];
Bits<5> rd = $encoding[11:7];
```

Operation

```
if (X[rs1] >= imm) {
    X[rd] = X[rs3];
}
```

Included in

Extension	Version
Xqci	~> 0.1.0
Xqcicm	~> 0.1.0

6.112. qc.mvlt

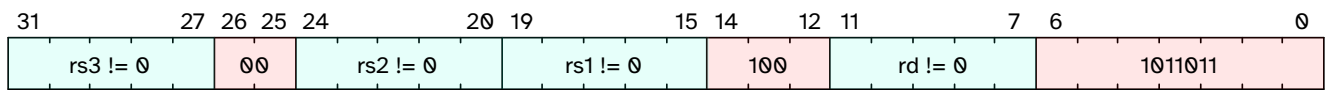
Synopsis

Conditional move if less than (Register)

Mnemonic

```
qc.mvlt rd, rs1, rs2, rs3
```

Encoding



Description

Move rs3 to rd if the value in rs1 is less than value rs2. Instruction encoded in R4 instruction format.

Decode Variables

```
Bits<5> rs1 = $encoding[19:15];
Bits<5> rs2 = $encoding[24:20];
Bits<5> rs3 = $encoding[31:27];
Bits<5> rd = $encoding[11:7];
```

Operation

```
if ($signed(X[rs1]) < $signed(X[rs2])) {
    X[rd] = X[rs3];
}
```

Included in

Extension	Version
Xqci	~> 0.1.0
Xqcicm	~> 0.1.0

6.113. qc.mvlti

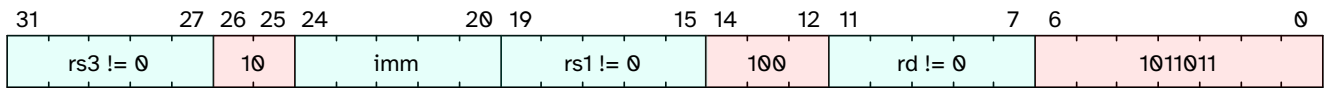
Synopsis

Conditional move if less than (Immediate)

Mnemonic

```
qc.mvlti rd, rs1, imm, rs3
```

Encoding



Description

Move `rs3` to `rd` if the value in `rs1` is less than value of `imm`. Instruction encoded in R4 instruction format.

Decode Variables

```
Bits<5> rs1 = $encoding[19:15];
Bits<5> imm = $encoding[24:20];
Bits<5> rs3 = $encoding[31:27];
Bits<5> rd = $encoding[11:7];
```

Operation

```
if ($signed(X[rs1]) < $signed(imm)) {
    X[rd] = X[rs3];
}
```

Included in

Extension	Version
Xqci	~> 0.1.0
Xqcicm	~> 0.1.0

6.114. qc.mvltu

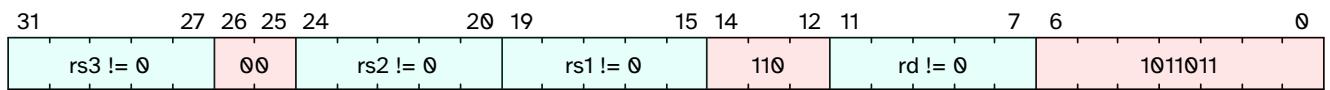
Synopsis

Conditional move if less than unsigned (Register)

Mnemonic

```
qc.mvltu rd, rs1, rs2, rs3
```

Encoding



Description

Move `rs3` to `rd` if the unsigned value in `rs1` is less than unsigned value `rs2`. Instruction encoded in R4 instruction format.

Decode Variables

```
Bits<5> rs1 = $encoding[19:15];
Bits<5> rs2 = $encoding[24:20];
Bits<5> rs3 = $encoding[31:27];
Bits<5> rd = $encoding[11:7];
```

Operation

```
if (X[rs1] < X[rs2]) {
    X[rd] = X[rs3];
}
```

Included in

Extension	Version
Xqci	~> 0.1.0
Xqcicm	~> 0.1.0

6.115. qc.mvltui

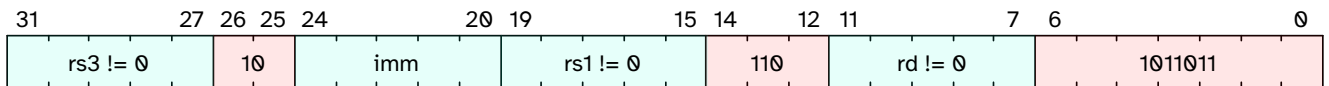
Synopsis

Conditional move if less than unsigned (Immediate)

Mnemonic

```
qc.mvltui rd, rs1, imm, rs3
```

Encoding



Description

Move `rs3` to `rd` if the unsigned value in `rs1` is less than unsigned value of `imm`. Instruction encoded in R4 instruction format.

Decode Variables

```
Bits<5> rs1 = $encoding[19:15];
Bits<5> imm = $encoding[24:20];
Bits<5> rs3 = $encoding[31:27];
Bits<5> rd = $encoding[11:7];
```

Operation

```
if (X[rs1] < imm) {
    X[rd] = X[rs3];
}
```

Included in

Extension	Version
Xqci	~> 0.1.0
Xqcicm	~> 0.1.0

6.116. qc.mvne

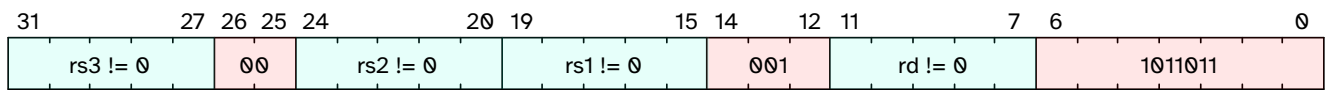
Synopsis

Conditional move if not equal (Register)

Mnemonic

```
qc.mvne rd, rs1, rs2, rs3
```

Encoding



Description

Move rs3 to rd if the value in rs1 is not equal to value rs2. Instruction encoded in R4 instruction format.

Decode Variables

```
Bits<5> rs1 = $encoding[19:15];
Bits<5> rs2 = $encoding[24:20];
Bits<5> rs3 = $encoding[31:27];
Bits<5> rd = $encoding[11:7];
```

Operation

```
if (X[rs1] != X[rs2]) {
    X[rd] = X[rs3];
}
```

Included in

Extension	Version
Xqci	~> 0.1.0
Xqcicm	~> 0.1.0

6.117. qc.mvnei

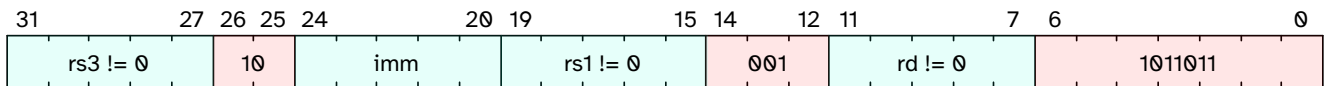
Synopsis

Conditional move if not equal (Immediate)

Mnemonic

```
qc.mvnei rd, rs1, imm, rs3
```

Encoding



Description

Move `rs3` to `rd` if the value in `rs1` is not equal to value `imm`. Instruction encoded in R4 instruction format.

Decode Variables

```
Bits<5> rs1 = $encoding[19:15];
Bits<5> imm = $encoding[24:20];
Bits<5> rs3 = $encoding[31:27];
Bits<5> rd = $encoding[11:7];
```

Operation

```
if ($signed(X[rs1]) != $signed(imm)) {
    X[rd] = X[rs3];
}
```

Included in

Extension	Version
Xqci	~> 0.1.0
Xqcicm	~> 0.1.0

6.118. qc.norm

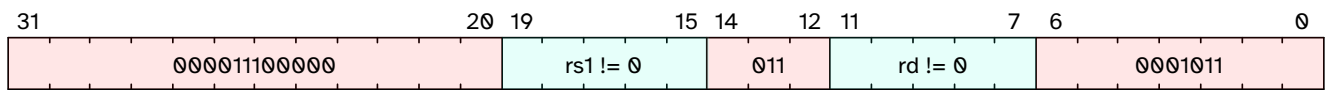
Synopsis

Signed Normalization

Mnemonic

```
qc.norm rd, rs1
```

Encoding



Description

Signed normalization of `rs1`. Exponent written in bits[7:0] of the result. Mantissa written in bits[31:8] of the result. Write result to `rd`. Instruction encoded in I instruction format.

Decode Variables

```
Bits<5> rs1 = $encoding[19:15];
Bits<5> rd = $encoding[11:7];
```

Operation

```
XReg clz = (xlen() - 1) - $signed(highest_set_bit(X[rs1]));
XReg clo = (xlen() - 1) - $signed(highest_set_bit(~X[rs1]));
XReg mnt = (X[rs1][31] == 1) ? (X[rs1] << (clo - 1)) : (X[rs1] << (clz - 1));
XReg exp = (X[rs1][31] == 1) ? (-(clo - 1)) : (-(clz - 1));
X[rd] = {mnt[31:8], exp[7:0]};
```

Included in

Extension	Version
Xqci	~> 0.1.0
Xqcia	~> 0.1.0

6.119. qc.normeu

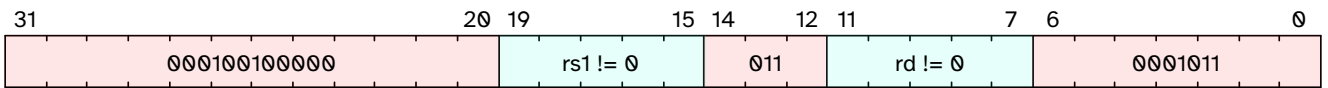
Synopsis

Unsigned Even Normalization

Mnemonic

```
qc.normeu rd, rs1
```

Encoding



Description

Unsigned even normalization of rs1 (exponent is even). Even exponent written in bits[7:0] of the result. Mantissa (based on even exponent) written in bits[31:8] of the result. Write result to rd. Instruction encoded in I instruction format.

Decode Variables

```
Bits<5> rs1 = $encoding[19:15];
Bits<5> rd = $encoding[11:7];
```

Operation

```
XReg clz = xlen() - 1 - $signed(highest_set_bit(X[rs1]) & ~1);
XReg mnt = (X[rs1] << (clz & ~1));
XReg exp = (-(clz & ~1));
X[rd] = {mnt[31:8], exp[7:0]};
```

Included in

Extension	Version
Xqci	~> 0.1.0
Xqcia	~> 0.1.0

6.120. qc.normu

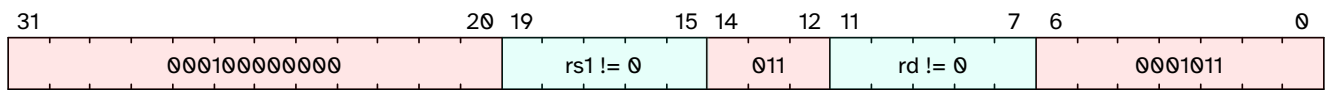
Synopsis

Unsigned Normalization

Mnemonic

```
qc.normu rd, rs1
```

Encoding



Description

Unsigned normalization of `rs1`. Exponent written in bits[7:0] of the result. Mantissa written in bits[31:8] of the result. Write result to `rd`. Instruction encoded in I instruction format.

Decode Variables

```
Bits<5> rs1 = $encoding[19:15];
Bits<5> rd = $encoding[11:7];
```

Operation

```
XReg clz = (xlen() - 1) - $signed(highest_set_bit(X[rs1]));
XReg mnt = (X[rs1] << clz);
XReg exp = (-clz);
X[rd] = {mnt[31:8], exp[7:0]};
```

Included in

Extension	Version
Xqci	~> 0.1.0
Xqcia	~> 0.1.0

6.121. qc.outw

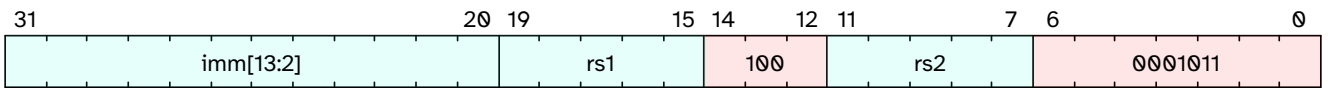
Synopsis

Output word to non-memory-mapped device

Mnemonic

```
qc.outw rs2, imm(rs1)
```

Encoding



Description

Output 32 bits of data from register `rs2` to a non-memory-mapped device. Such devices have own address space, unrelated to memory map. Device space address formed by adding `rs1` to to a unsigned offset `imm`. Instruction encoded in I instruction format.

Decode Variables

```
Bits<14> imm = {$encoding[31:20], 2'd0};
Bits<5> rs1 = $encoding[19:15];
Bits<5> rs2 = $encoding[11:7];
```

Operation

```
XReg device_address = X[rs1] + imm;
write_device_32(device_address, X[rs2]);
```

Included in

Extension	Version
Xqci	~> 0.1.0
Xqciio	~> 0.1.0

6.122. qc.pcoredump

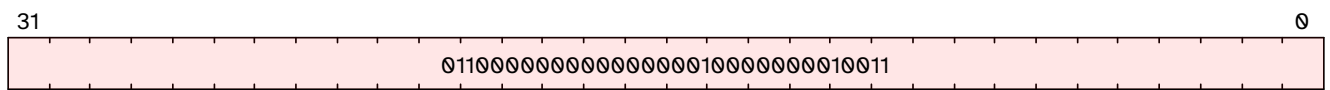
Synopsis

Print core dump pseudo-instruction (hint) working only in simulation environment

Mnemonic

```
qc.pcoredump
```

Encoding



Description

The print core dump instruction calls simulation environment with no explicit arguments. Implicit arguments are all general purpose registers and CSRs. Simulation environment expected to print the core dump on its console or standard output. The core dump format and content are defined by simulation environment. Instruction encoded in I instruction format.

Decode Variables

qc.pcoredump has no decode variables.

Operation

```
XReg func = 8;
XReg arg = 0;
iss_syscall(func, arg);
```

Included in

Extension	Version
Xqci	~> 0.1.0
Xqcisim	~> 0.1.0

6.123. qc.pexit

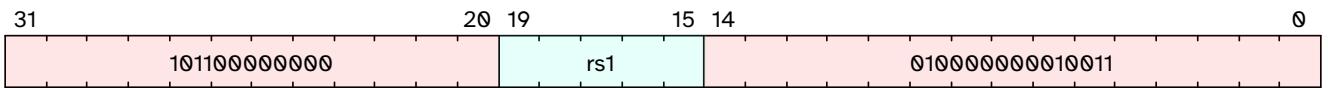
Synopsis

Exit call with register argument pseudo-instruction (hint) working only in simulation environment

Mnemonic

```
qc.pexit  rs1
```

Encoding



Description

The Exit call instruction calls simulation environment with unsigned `rs1` explicit argument. Simulation environment is expected to complete its execution and return to the system with exit code provided in `rs1`. Instruction encoded in I instruction format.

Decode Variables

```
Bits<5> rs1 = $encoding[19:15];
```

Operation

```
XReg func = 12;
XReg arg = X[rs1];
iss_syscall(func, arg);
```

Included in

Extension	Version
Xqci	~> 0.1.0
Xqcisim	~> 0.1.0

6.124. qc.ppreg

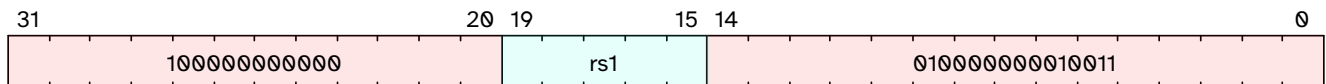
Synopsis

Print register pseudo-instruction (hint) working only in simulation environment

Mnemonic

```
qc.ppreg  rs1
```

Encoding



Description

The print register instruction calls simulation environment with `rs1` explicit argument. Simulation environment expected to print the register value on its console or standard output. Instruction encoded in I instruction format.

Decode Variables

```
Bits<5> rs1 = $encoding[19:15];
```

Operation

```
XReg func = 2;
XReg arg = X[rs1];
iss_syscall(func, arg);
```

Included in

Extension	Version
Xqci	~> 0.1.0
Xqcisim	~> 0.1.0

6.125. qc.ppregs

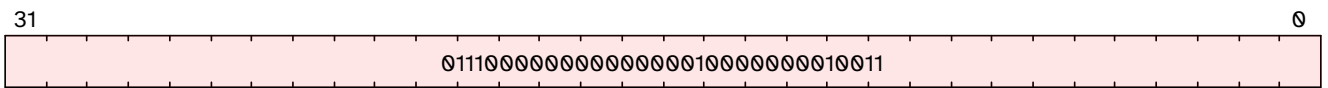
Synopsis

Print all registers pseudo-instruction (hint) working only in simulation environment

Mnemonic

```
qc.ppregs
```

Encoding



Description

The print registers instruction calls simulation environment with no explicit arguments. Implicit arguments are all general purpose registers. Simulation environment expected to print the all registers value on its console or standard output. Instruction encoded in I instruction format.

Decode Variables

qc.ppregs has no decode variables.

Operation

```
XReg func = 3;
XReg arg = 0;
iss_syscall(func, arg);
```

Included in

Extension	Version
Xqci	~> 0.1.0
Xqcisim	~> 0.1.0

6.126. qc.pputc

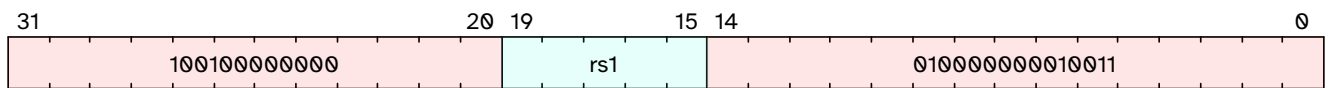
Synopsis

Print character passed in register argument pseudo-instruction (hint) working only in simulation environment

Mnemonic

```
qc.pputc  rs1
```

Encoding



Description

The print character instruction calls simulation environment with `rs1` explicit argument. Simulation environment expected to print the character on its console or standard output. Instruction encoded in I instruction format.

Decode Variables

```
Bits<5> rs1 = $encoding[19:15];
```

Operation

```
XReg func = 4;
XReg arg = X[rs1];
iss_syscall(func, arg);
```

Included in

Extension	Version
Xqci	~> 0.1.0
Xqcisim	~> 0.1.0

6.127. qc.pputci

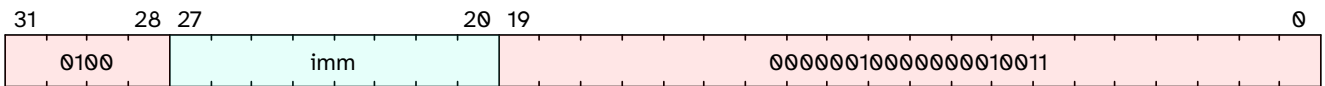
Synopsis

Print character passed in the immediate argument pseudo-instruction (hint) working only in simulation environment

Mnemonic

```
qc.pputci imm
```

Encoding



Description

The print character instruction calls simulation environment with unsigned `imm` explicit argument. Simulation environment expected to print the 8-bit character on its console or standard output. Instruction encoded in I instruction format.

Decode Variables

```
Bits<8> imm = $encoding[27:20];
```

Operation

```
XReg func = 5;
XReg arg = imm;
iss_syscall(func, arg);
```

Included in

Extension	Version
Xqci	~> 0.1.0
Xqcisim	~> 0.1.0

6.128. qc.pputs

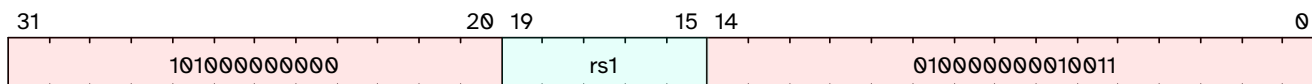
Synopsis

Print string pseudo-instruction (hint) working only in simulation environment

Mnemonic

```
qc.pputs  rs1
```

Encoding



Description

The print string instruction calls simulation environment with `rs1` explicit argument. The argument assumed to be pointer to the string in target simulated memory. Simulation environment expected to print the string on its console or standard output. Instruction encoded in I instruction format.

Decode Variables

```
Bits<5> rs1 = $encoding[19:15];
```

Operation

```
XReg func = 6;  
XReg arg = X[rs1];  
iss_syscall(func, arg);
```

Included in

Extension	Version
Xqci	~> 0.1.0
Xqcisim	~> 0.1.0

6.129. qc.psyscall

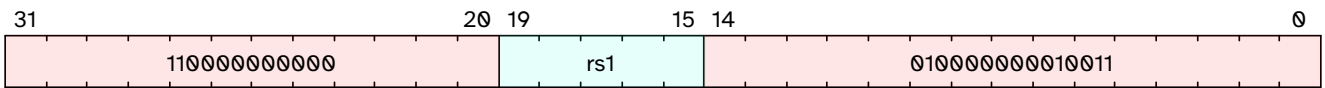
Synopsis

System call with register argument pseudo-instruction (hint) working only in simulation environment

Mnemonic

```
qc.psyscall  rs1
```

Encoding



Description

The System call instruction calls simulation environment with unsigned `rs1` explicit argument. Additional implicit arguments defined by simulation environment implementation. Instruction is used to call simulator to execute function according to the argument. Instruction encoded in I instruction format.

Decode Variables

```
Bits<5> rs1 = $encoding[19:15];
```

Operation

```
XReg func = 10;
XReg arg = X[rs1];
iss_syscall(func, arg);
```

Included in

Extension	Version
Xqci	~> 0.1.0
Xqcisim	~> 0.1.0

6.130. qc.psyscalli

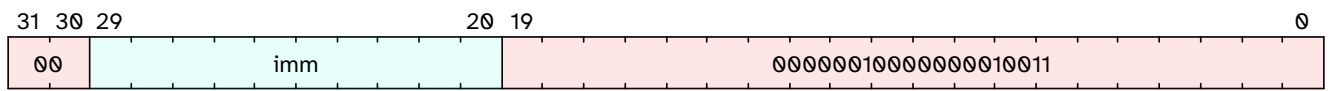
Synopsis

System call with immediate argument pseudo-instruction (hint) working only in simulation environment

Mnemonic

```
qc.psyscalli imm
```

Encoding



Description

The System call instruction calls simulation environment with unsigned `imm` explicit argument. Additional implicit arguments defined by simulation environment implementation. Instruction is used to call simulator to execute function according to the argument. Instruction encoded in I instruction format.

Decode Variables

```
Bits<10> imm = $encoding[29:20];
```

Operation

```
XReg func = 11;
XReg arg = imm;
iss_syscall(func, arg);
```

Included in

Extension	Version
Xqci	~> 0.1.0
Xqcisim	~> 0.1.0

6.131. qc.selecteqi

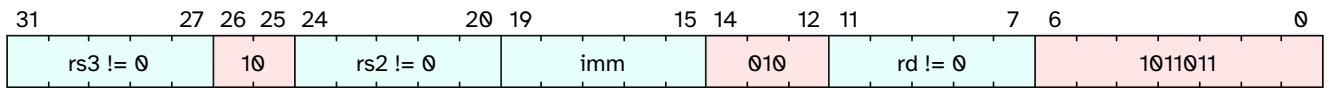
Synopsis

Select load immediate or register if equal (Immediate)

Mnemonic

```
qc.selecteqi rd, imm, rs2, rs3
```

Encoding



Description

Move rs2 to rd if the value in rd is equal to value imm, move rs3 to rd otherwise. Instruction encoded in R4 instruction format.

Decode Variables

```
Bits<5> imm = $encoding[19:15];
Bits<5> rs2 = $encoding[24:20];
Bits<5> rs3 = $encoding[31:27];
Bits<5> rd = $encoding[11:7];
```

Operation

```
if ($signed(X[rd]) == $signed(imm)) {
    X[rd] = X[rs2];
} else {
    X[rd] = X[rs3];
}
```

Included in

Extension	Version
Xqci	~> 0.1.0
Xqcics	~> 0.1.0

6.132. qc.selectieq

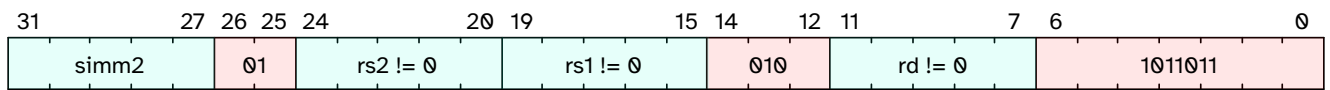
Synopsis

Select load immediate or register if equal (Register)

Mnemonic

```
qc.selectieq rd, rs1, rs2, simm2
```

Encoding



Description

Move rs2 to rd if the value in rd is equal to value rs1, move simm2 to rd otherwise. Instruction encoded in R4 instruction format.

Decode Variables

```
Bits<5> rs1 = $encoding[19:15];
Bits<5> rs2 = $encoding[24:20];
Bits<5> simm2 = $encoding[31:27];
Bits<5> rd = $encoding[11:7];
```

Operation

```
if (X[rd] == X[rs1]) {
    X[rd] = X[rs2];
} else {
    X[rd] = sext(simm2, 5);
}
```

Included in

Extension	Version
Xqci	~> 0.1.0
Xqcics	~> 0.1.0

6.133. qc.selectieqi

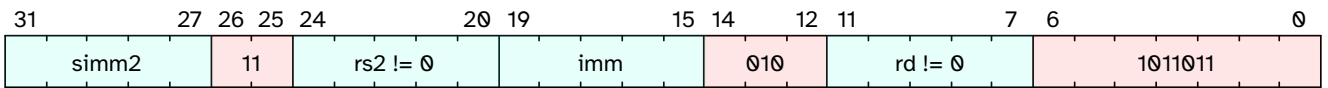
Synopsis

Select load immediate or register if equal (Immediate)

Mnemonic

```
qc.selectieqi rd, imm, rs2, simm2
```

Encoding



Description

Move rs2 to rd if the value in rd is equal to value imm, move simm2 to rd otherwise. Instruction encoded in R4 instruction format.

Decode Variables

```
Bits<5> imm = $encoding[19:15];
Bits<5> rs2 = $encoding[24:20];
Bits<5> simm2 = $encoding[31:27];
Bits<5> rd = $encoding[11:7];
```

Operation

```
if ($signed(X[rd]) == $signed(imm)) {
    X[rd] = X[rs2];
} else {
    X[rd] = sext(simm2, 5);
}
```

Included in

Extension	Version
Xqci	~> 0.1.0
Xqcics	~> 0.1.0

6.134. qc.selectiieq

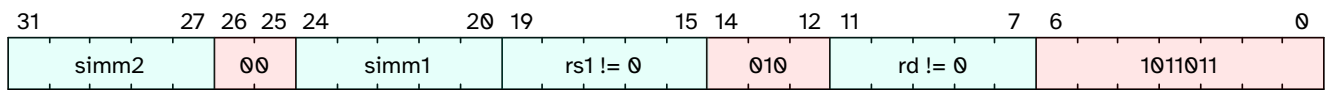
Synopsis

Select load immediate if equal (Register)

Mnemonic

```
qc.selectiieq rd, rs1, simm1, simm2
```

Encoding



Description

Move simm1 to rd if the value in rd is equal to value rs1, move simm2 to rd otherwise. Instruction encoded in R4 instruction format.

Decode Variables

```
Bits<5> rs1 = $encoding[19:15];
Bits<5> simm1 = $encoding[24:20];
Bits<5> simm2 = $encoding[31:27];
Bits<5> rd = $encoding[11:7];
```

Operation

```
if ($signed(X[rd]) == $signed(X[rs1])) {
    X[rd] = sext(simm1, 5);
} else {
    X[rd] = sext(simm2, 5);
}
```

Included in

Extension	Version
Xqci	~> 0.1.0
Xqcics	~> 0.1.0

6.135. qc.selectiine

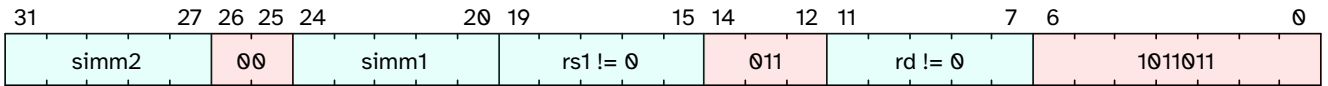
Synopsis

Select load immediate if not equal (Register)

Mnemonic

```
qc.selectiine rd, rs1, simm1, simm2
```

Encoding



Description

Move simm1 to rd if the value in rd is not equal to value rs1, move simm2 to rd otherwise. Instruction encoded in R4 instruction format.

Decode Variables

```
Bits<5> rs1 = $encoding[19:15];
Bits<5> simm1 = $encoding[24:20];
Bits<5> simm2 = $encoding[31:27];
Bits<5> rd = $encoding[11:7];
```

Operation

```
if ($signed(X[rd]) != $signed(X[rs1])) {
    X[rd] = sext(simm1, 5);
} else {
    X[rd] = sext(simm2, 5);
}
```

Included in

Extension	Version
Xqci	~> 0.1.0
Xqcics	~> 0.1.0

6.136. qc.selectine

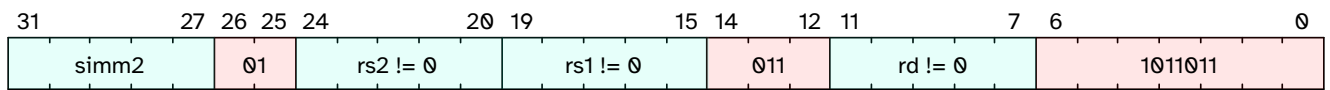
Synopsis

Select load immediate or register if not equal (Register)

Mnemonic

```
qc.selectine rd, rs1, rs2, simm2
```

Encoding



Description

Move rs2 to rd if the value in rd is not equal to value rs1, move simm2 to rd otherwise. Instruction encoded in R4 instruction format.

Decode Variables

```
Bits<5> rs1 = $encoding[19:15];
Bits<5> rs2 = $encoding[24:20];
Bits<5> simm2 = $encoding[31:27];
Bits<5> rd = $encoding[11:7];
```

Operation

```
if ($signed(X[rd]) != $signed(X[rs1])) {
    X[rd] = X[rs2];
} else {
    X[rd] = sext(simm2, 5);
}
```

Included in

Extension	Version
Xqci	~> 0.1.0
Xqcics	~> 0.1.0

6.137. qc.selectinei

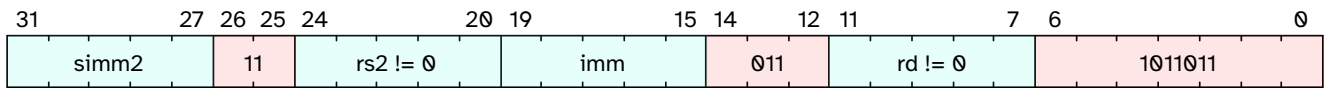
Synopsis

Select load immediate or register if not equal (Immediate)

Mnemonic

```
qc.selectinei rd, imm, rs2, simm2
```

Encoding



Description

Move rs2 to rd if the value in rd is not equal to value imm, move simm2 to rd otherwise. Instruction encoded in R4 instruction format.

Decode Variables

```
Bits<5> imm = $encoding[19:15];
Bits<5> rs2 = $encoding[24:20];
Bits<5> simm2 = $encoding[31:27];
Bits<5> rd = $encoding[11:7];
```

Operation

```
if ($signed(X[rd]) != $signed(imm)) {
    X[rd] = X[rs2];
} else {
    X[rd] = sext(simm2, 5);
}
```

Included in

Extension	Version
Xqci	~> 0.1.0
Xqcics	~> 0.1.0

6.138. qc.selectnei

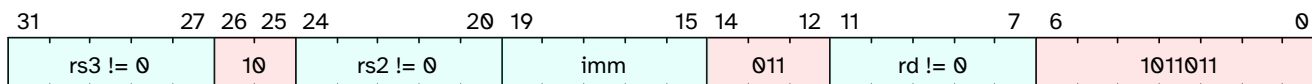
Synopsis

Select load immediate or register if not equal (Immediate)

Mnemonic

```
qc.selectnei rd, imm, rs2, rs3
```

Encoding



Description

Move rs2 to rd if the value in rd is not equal to value imm, move rs3 to rd otherwise. Instruction encoded in R4 instruction format.

Decode Variables

```
Bits<5> imm = $encoding[19:15];
Bits<5> rs2 = $encoding[24:20];
Bits<5> rs3 = $encoding[31:27];
Bits<5> rd = $encoding[11:7];
```

Operation

```
if ($signed(X[rd]) != $signed(imm)) {
    X[rd] = X[rs2];
} else {
    X[rd] = X[rs3];
}
```

Included in

Extension	Version
Xqci	~> 0.1.0
Xqcics	~> 0.1.0

6.139. qc.setinti

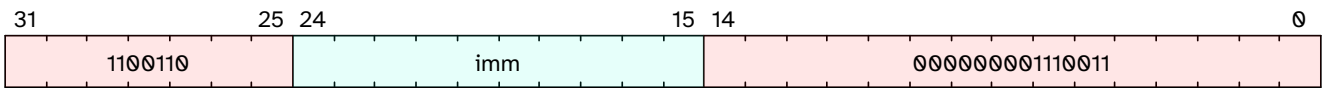
Synopsis

Set interrupt (Immediate)

Mnemonic

```
qc.setinti imm
```

Encoding



Description

Set interrupt, interrupt number is in `imm` (0 - 1023). Instruction encoded in R instruction format.

Decode Variables

```
Bits<10> imm = $encoding[24:15];
```

Operation

```
Bits<12> MCLICIP0_ADDR = CSR[qc.mclicip0].address();
XReg idx = imm / 32;
XReg bit = imm % 32;
XReg pre_csr = CSR[MCLICIP0_ADDR + idx].sw_read();
CSR[MCLICIP0_ADDR + idx].sw_write(pre_csr | (1 << bit));
```

Included in

Extension	Version
Xqci	~> 0.1.0
Xqciint	~> 0.1.0

6.140. qc.setwm

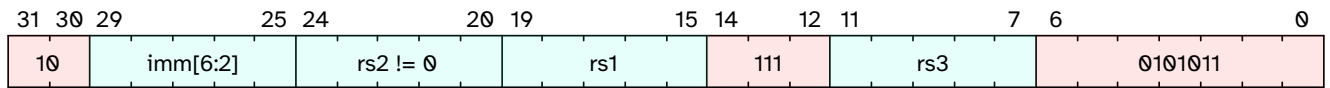
Synopsis

Set word multiple (Register)

Mnemonic

```
qc.setwm  rs3, rs2, imm(rs1)
```

Encoding



Description

Stores the value of `rs3` multiple times into the address starting at `(rs1 + imm)`. The number of writes is in `rs2` bits `[4:0]` (`0..31`). Instruction encoded in R instruction format.

Decode Variables

```
Bits<7> imm = {$encoding[29:25], 2'd0};
Bits<5> rs1 = $encoding[19:15];
Bits<5> rs2 = $encoding[24:20];
Bits<5> rs3 = $encoding[11:7];
```

Operation

```
XReg vaddr = X[rs1] + imm;
Bits<32> write_value = X[rs3][31:0];
XReg num_words = X[rs2][4:0];
for (U32 i = 0; i < num_words; i++) {
    write_memory<32>(vaddr, write_value, $encoding);
    vaddr = vaddr + 4;
}
```

Included in

Extension	Version
Xqci	~> 0.1.0
Xqcilsm	~> 0.1.0

6.141. qc.setwmi

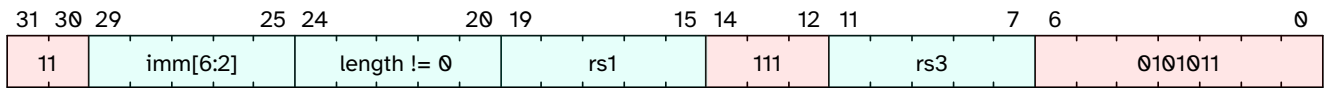
Synopsis

Set word multiple (Immediate)

Mnemonic

```
qc.setwmi rs3, length, imm(rs1)
```

Encoding



Description

Stores the value of `rs3` multiple times into the address starting at `(rs1 + imm)`. The number of writes is in `length`. Instruction encoded in R instruction format.

Decode Variables

```
Bits<7> imm = {$encoding[29:25], 2'd0};
Bits<5> rs1 = $encoding[19:15];
Bits<5> length = $encoding[24:20];
Bits<5> rs3 = $encoding[11:7];
```

Operation

```
XReg vaddr = X[rs1] + imm;
Bits<32> write_value = X[rs3][31:0];
for (U32 i = 0; i < length; i++) {
    write_memory<32>(vaddr, write_value, $encoding);
    vaddr = vaddr + 4;
}
```

Included in

Extension	Version
Xqci	~> 0.1.0
Xqcilsm	~> 0.1.0

6.142. qc.shladd

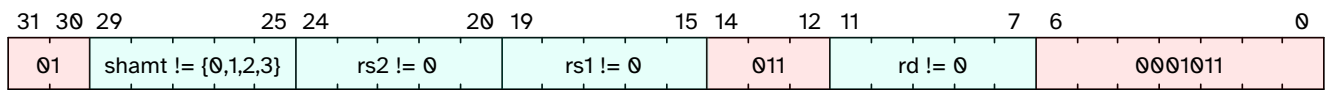
Synopsis

Shift left and add (immediate)

Mnemonic

```
qc.shladd rd, rs1, rs2, shamt
```

Encoding



Description

Left shift *rs1* by *shamt* and add the value in *rs2*. Instruction encoded in R instruction format.

Decode Variables

```
Bits<5> shamt = $encoding[29:25];
Bits<5> rs2 = $encoding[24:20];
Bits<5> rs1 = $encoding[19:15];
Bits<5> rd = $encoding[11:7];
```

Operation

```
X[rd] = (X[rs1] << shamt) + X[rs2];
```

Included in

Extension	Version
Xqci	>= 0.2
Xqciac	~> 0.1.0

6.143. qc.shlsat

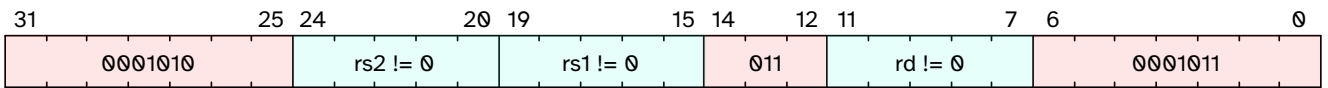
Synopsis

Saturating signed left shift

Mnemonic

```
qc.shlsat rd, rs1, rs2
```

Encoding



Description

Left shift `rs1` by the value of `rs2`, and saturate the signed result. Instruction encoded in R instruction format.

Decode Variables

```
Bits<5> rs1 = $encoding[19:15];
Bits<5> rs2 = $encoding[24:20];
Bits<5> rd = $encoding[11:7];
```

Operation

```
Bits<{1'b0, MXLEN} * 2> sext_double_width_rs1 = {{MXLEN{X[rs1][xlen() - 1]}},
X[rs1]};
Bits<{1'b0, MXLEN} * 2> shifted_value = sext_double_width_rs1 << X[rs2][4:0];
XReg most_negative_number = 1 << (xlen() - 1);
XReg most_positive_number = (1 << (xlen() - 1)) - 1;
if ($signed(shifted_value) < $signed(most_negative_number)) {
    X[rd] = most_negative_number;
} else if ($signed(shifted_value) > $signed(most_positive_number)) {
    X[rd] = most_positive_number;
} else {
    X[rd] = shifted_value;
}
```

Included in

Extension	Version
Xqci	~> 0.1.0
Xqcia	~> 0.1.0

6.144. qc.shlusat

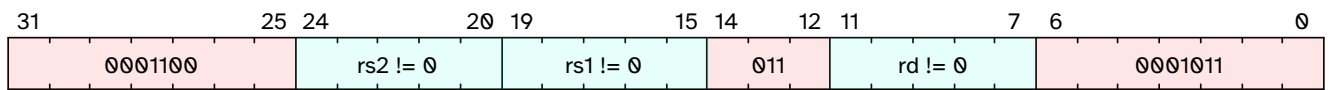
Synopsis

Saturating unsigned left shift

Mnemonic

```
qc.shlusat rd, rs1, rs2
```

Encoding



Description

Left shift `rs1` by the value of `rs2`, and saturate the unsigned result. Instruction encoded in R instruction format.

Decode Variables

```
Bits<5> rs1 = $encoding[19:15];
Bits<5> rs2 = $encoding[24:20];
Bits<5> rd = $encoding[11:7];
```

Operation

```
Bits<{1'b0, MXLEN} * 2> sext_double_width_rs1 = {{MXLEN{X[rs1][xlen() - 1]}},
X[rs1]};
Bits<{1'b0, MXLEN} * 2> shifted_value = sext_double_width_rs1 << X[rs2][4:0];
XReg largest_unsigned_value = {MXLEN{1'b1}};
if (shifted_value > largest_unsigned_value) {
    X[rd] = largest_unsigned_value;
} else {
    X[rd] = shifted_value;
}
```

Included in

Extension	Version
Xqci	~> 0.1.0
Xqcia	~> 0.1.0

6.145. qc.srb

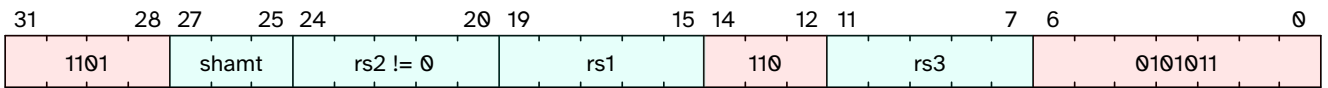
Synopsis

Store indexed byte

Mnemonic

```
qc.srb  rs3, rs1, rs2, shamt
```

Encoding



Description

Store 8 bits of data from register `rs3` to an address formed by adding `rs1` to `rs2`, shifted by `shamt`. Instruction encoded in R instruction format.

Decode Variables

```
Bits<3> shamt = $encoding[27:25];
Bits<5> rs1 = $encoding[19:15];
Bits<5> rs2 = $encoding[24:20];
Bits<5> rs3 = $encoding[11:7];
```

Operation

```
XReg virtual_address = X[rs1] + (X[rs2] << shamt);
write_memory<8>(virtual_address, X[rs3][7:0], $encoding);
```

Included in

Extension	Version
Xqci	~> 0.1.0
Xqcisls	~> 0.1.0

6.146. qc.srh

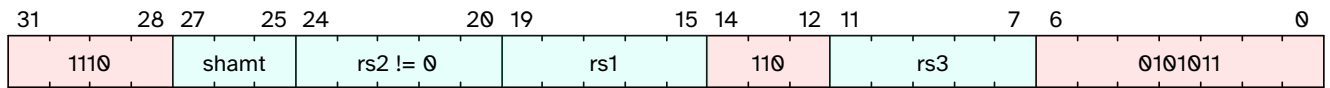
Synopsis

Store indexed halfword

Mnemonic

```
qc.srh  rs3, rs1, rs2, shamt
```

Encoding



Description

Store 16 bits of data from register `rs3` to an address formed by adding `rs1` to `rs2`, shifted by `shamt`. Instruction encoded in R instruction format.

Decode Variables

```
Bits<3> shamt = $encoding[27:25];
Bits<5> rs1 = $encoding[19:15];
Bits<5> rs2 = $encoding[24:20];
Bits<5> rs3 = $encoding[11:7];
```

Operation

```
XReg virtual_address = X[rs1] + (X[rs2] << shamt);
write_memory<16>(virtual_address, X[rs3][15:0], $encoding);
```

Included in

Extension	Version
Xqci	~> 0.1.0
Xqcisls	~> 0.1.0

6.147. qc.srw

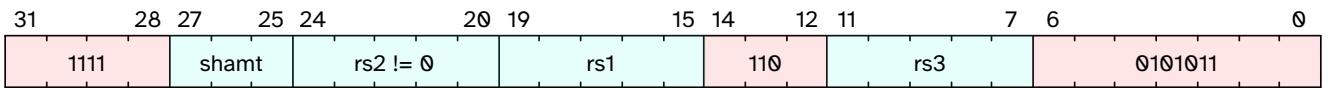
Synopsis

Store indexed word

Mnemonic

```
qc.srw  rs3, rs1, rs2, shamt
```

Encoding



Description

Store 32 bits of data from register `rs3` to an address formed by adding `rs1` to `rs2`, shifted by `shamt`. Instruction encoded in R instruction format.

Decode Variables

```
Bits<3> shamt = $encoding[27:25];
Bits<5> rs1 = $encoding[19:15];
Bits<5> rs2 = $encoding[24:20];
Bits<5> rs3 = $encoding[11:7];
```

Operation

```
XReg virtual_address = X[rs1] + (X[rs2] << shamt);
write_memory<32>(virtual_address, X[rs3][31:0], $encoding);
```

Included in

Extension	Version
Xqci	~> 0.1.0
Xqcisls	~> 0.1.0

6.148. qc.subsat

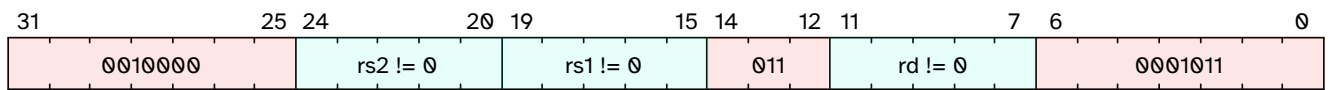
Synopsis

Saturating signed subtraction

Mnemonic

```
qc.subsat rd, rs1, rs2
```

Encoding



Description

Subtract signed values `rs1` and `rs2`, saturate the signed result, and write to `rd`. Instruction encoded in R instruction format.

Decode Variables

```
Bits<5> rs1 = $encoding[19:15];
Bits<5> rs2 = $encoding[24:20];
Bits<5> rd = $encoding[11:7];
```

Operation

```
XReg result = X[rs1] - X[rs2];
XReg most_negative_number = 1 << (xlen() - 1);
XReg most_positive_number = (1 << (xlen() - 1)) - 1;
if (X[rs1][xlen() - 1] != X[rs2][xlen() - 1]) {
    if (result[xlen() - 1] != X[rs1][xlen() - 1]) {
        if ($signed(X[rs1]) < 0) {
            result = most_negative_number;
        } else {
            result = most_positive_number;
        }
    }
}
X[rd] = result;
```

Included in

Extension	Version
Xqci	~> 0.1.0
Xqcia	~> 0.1.0

6.149. qc.subusat

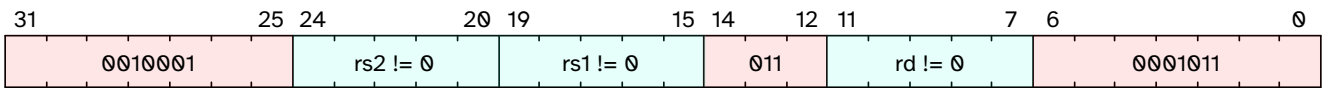
Synopsis

Saturating unsigned subtraction

Mnemonic

```
qc.subusat rd, rs1, rs2
```

Encoding



Description

Subtract unsigned values `rs1` and `rs2`, saturate the unsigned result, and write to `rd`. Instruction encoded in R instruction format.

Decode Variables

```
Bits<5> rs1 = $encoding[19:15];
Bits<5> rs2 = $encoding[24:20];
Bits<5> rd = $encoding[11:7];
```

Operation

```
X[rd] = (X[rs1] < X[rs2]) ? 0 : X[rs1] - X[rs2];
```

Included in

Extension	Version
Xqci	~> 0.1.0
Xqcia	~> 0.1.0

6.150. qc.swm

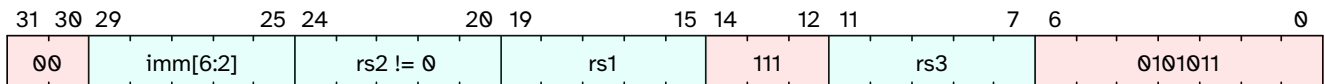
Synopsis

Store word multiple

Mnemonic

```
qc.swm  rs3, rs2, imm(rs1)
```

Encoding



Description

Stores multiple words from the registers starting at `rs3` to the address starting at `(rs1 + imm)`. The number of words is in `rs2`. Instruction encoded in R instruction format.

Decode Variables

```
Bits<7> imm = {$encoding[29:25], 2'd0};
Bits<5> rs1 = $encoding[19:15];
Bits<5> rs2 = $encoding[24:20];
Bits<5> rs3 = $encoding[11:7];
```

Operation

```
XReg vaddr = X[rs1] + imm;
XReg num_words = X[rs2];
raise(ExceptionCode::IllegalInstruction, effective_ldst_mode(), $encoding) if
(rs3 + num_words) > 32;
for (U32 i = 0; i < num_words; i++) {
    write_memory<32>(vaddr, X[rs3 + i], $encoding);
    vaddr = vaddr + 4;
}
```

Included in

Extension	Version
Xqci	~> 0.1.0
Xqcilsm	~> 0.1.0

6.151. qc.swmi

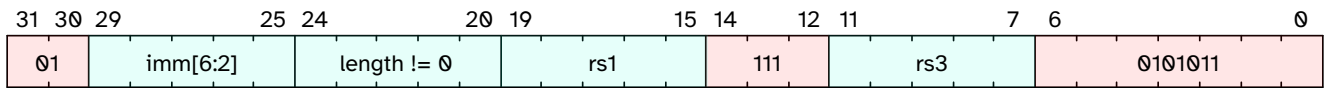
Synopsis

Store word multiple (immediate)

Mnemonic

```
qc.swmi  rs3, length, imm(rs1)
```

Encoding



Description

Stores multiple words from the registers starting at `rs3` to the address starting at `(rs1 + imm)`. The number of words is in `length` immediate. Instruction encoded in R instruction format.

Decode Variables

```
Bits<7> imm = {$encoding[29:25], 2'd0};
Bits<5> rs1 = $encoding[19:15];
Bits<5> length = $encoding[24:20];
Bits<5> rs3 = $encoding[11:7];
```

Operation

```
XReg vaddr = X[rs1] + imm;
raise(ExceptionCode::IllegalInstruction, effective_ldst_mode(), $encoding) if
(rs3 + length) > 32;
for (U32 i = 0; i < length; i++) {
    write_memory<32>(vaddr, X[rs3 + i], $encoding);
    vaddr = vaddr + 4;
}
```

Included in

Extension	Version
Xqci	~> 0.1.0
Xqcilsm	~> 0.1.0

6.152. qc.sync

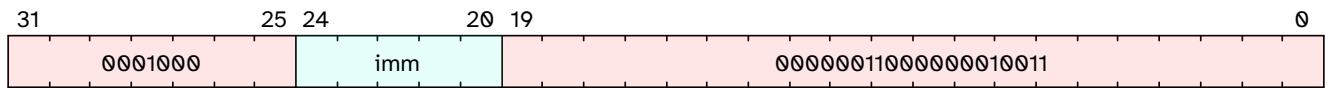
Synopsis

Non-memory-mapped device read-after-write synchronization to completion

Mnemonic

```
qc.sync  imm
```

Encoding



Description

This synchronization instruction delays execution till last read (input) from non-memory-mapped device transactions are completed for specified devices. As well, instruction implies fence.tso functionality for all memory accesses. Immediate argument is 5-bit bitmask field that specifies up to five pre-defined devices. Instruction encoded in I instruction format.

Decode Variables

```
Bits<5> imm = $encoding[24:20];
```

Operation

```
fence_tso();
sync_read_after_write_device(true, imm);
```

Included in

Extension	Version
Xqci	~> 0.1.0
Xqcisync	~> 0.1.0

6.153. qc.syncr

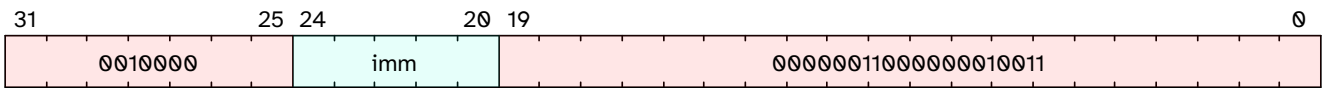
Synopsis

Non-memory-mapped device read-after-write synchronization

Mnemonic

```
qc.syncr  imm
```

Encoding



Description

This synchronization instruction delays execution till last read (input) from non-memory-mapped device transactions are started for specified devices. As well, instruction implies fence.tso functionality for all memory accesses. Immediate argument is 5-bit bitmask field that specifies up to five pre-defined devices. Instruction encoded in I instruction format.

Decode Variables

```
Bits<5> imm = $encoding[24:20];
```

Operation

```
fence_tso();
sync_read_after_write_device(false, imm);
```

Included in

Extension	Version
Xqci	~> 0.1.0
Xqcisync	~> 0.1.0

6.154. qc.syncwf

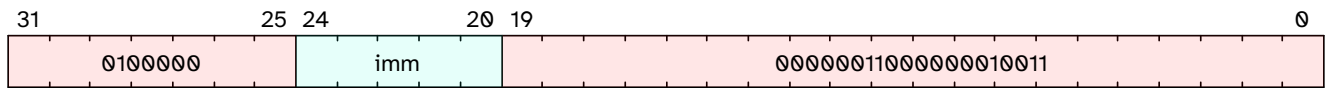
Synopsis

Non-memory-mapped device write-after-read synchronization

Mnemonic

```
qc.syncwf  imm
```

Encoding



Description

This synchronization instruction delays execution till last write (output) to non-memory-mapped device transactions are started for specified devices. As well, instruction implies fence.tso functionality for all memory accesses. Immediate argument is 5-bit bitmask field that specifies up to five pre-defined devices. Instruction encoded in I instruction format.

Decode Variables

```
Bits<5> imm = $encoding[24:20];
```

Operation

```
fence_tso();
sync_write_after_read_device(false, imm);
```

Included in

Extension	Version
Xqci	~> 0.1.0
Xqcisync	~> 0.1.0

6.155. qc.syncwl

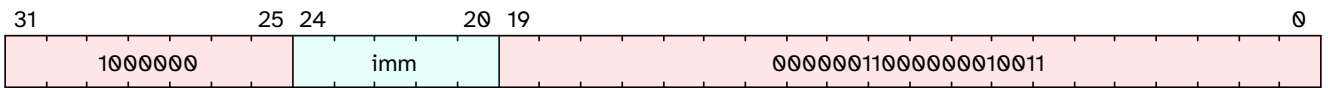
Synopsis

Non-memory-mapped device write-after-read synchronization to completion

Mnemonic

```
qc.syncwl imm
```

Encoding



Description

This synchronization instruction delays execution till last write (output) to non-memory-mapped device transactions are completed for specified devices. As well, instruction implies fence.tso functionality for all memory accesses. Immediate argument is 5-bit bitmask field that specifies up to five pre-defined devices. Instruction encoded in I instruction format.

Decode Variables

```
Bits<5> imm = $encoding[24:20];
```

Operation

```
fence_tso();
sync_write_after_read_device(true, imm);
```

Included in

Extension	Version
Xqci	~> 0.1.0
Xqcisync	~> 0.1.0

6.156. qc.wrap

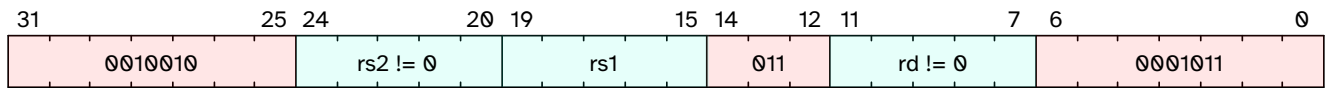
Synopsis

Wraparound (Register)

Mnemonic

```
qc.wrap rd, rs1, rs2
```

Encoding



Description

If $rs1 \geq rs2$ perform subtraction between $rs1$ and $rs2$. If $rs1 < 0$, perform addition between $rs1$ and $rs2$, else, select $rs1$. The result is stored in rd . Instruction encoded in R instruction format.

Decode Variables

```
Bits<5> rs1 = $encoding[19:15];
Bits<5> rs2 = $encoding[24:20];
Bits<5> rd = $encoding[11:7];
```

Operation

```
XReg rs1_value = X[rs1];
X[rd] = ($signed(rs1_value) >= $signed(X[rs2])) ? rs1_value - X[rs2] :
(($signed(rs1_value) < 0) ? (rs1_value + X[rs2]) : rs1_value);
```

Included in

Extension	Version
Xqci	~> 0.1.0
Xqcia	~> 0.1.0

6.157. qc.wrapi

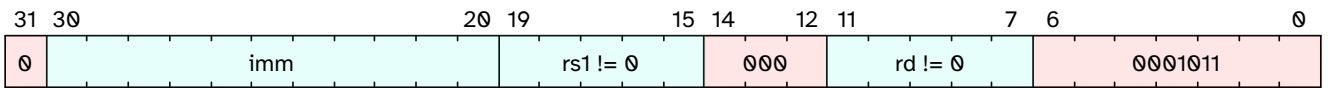
Synopsis

Wraparound (Unsigned Immediate)

Mnemonic

```
qc.wrapi rd, rs1, imm
```

Encoding



Description

If `rs1 >= imm` perform subtraction between `rs1` and `imm`. If `rs1 < 0`, perform addition between `rs1` and `imm`, else, select `rs1`. The result is stored in `rd`. Instruction encoded in I instruction format. The `imm` is an unsigned immediate.

Decode Variables

```
Bits<11> imm = $encoding[30:20];
Bits<5> rs1 = $encoding[19:15];
Bits<5> rd = $encoding[11:7];
```

Operation

```
XReg rs1_value = X[rs1];
X[rd] = ($signed(rs1_value) >= imm) ? rs1_value - imm : (($signed(rs1_value) < 0) ? ($signed(rs1_value) + imm) : rs1_value);
```

Included in

Extension	Version
Xqci	~> 0.1.0
Xqcia	~> 0.1.0

Chapter 7. IDL Functions

7.1. abort_current_instruction (builtin)

Abort the current instruction, and start refetching from \$pc.

Return Type	void
Arguments	

7.2. access_check

Checks if the physical address paddr is able to access memory, and raises the appropriate exception if not.

Return Type	void
Arguments	Bits<PHYS_ADDR_WIDTH> paddr, U32 access_size, XReg vaddr, MemoryOperation type, ExceptionCode fault_type, PrivilegeMode from_mode

```

if (paddr > 1 << PHYS_ADDR_WIDTH) - access_size {
    raise(fault_type, from_mode, vaddr);
}
if (implemented?(ExtensionName::Smpmp)) {
    if (!pmp_check(paddr[PHYS_ADDR_WIDTH - 1:0], access_size, type)) {
        raise(fault_type, from_mode, vaddr);
    }
}

```

7.3. assert (builtin)

Assert that a condition is true. Failure represents an error in the IDL model.

Return Type	void
Arguments	Boolean test, String message

7.4. cached_translation (generated)

Possibly returns a cached translation result matching vaddr.

CachedTranslationResult contains a Boolean 'valid' field. If valid, 'result' is a usable translation. Otherwise, the cache lookup failed.

Return Type	CachedTranslationResult
Arguments	XReg vaddr, MemoryOperation op

7.5. creg2reg

Maps a C register index (e.g., `rs1'` in the specification) to an X register index. From the specification:

Table 1. Registers specified by the three-bit `rs1'`, `rs2'`, and `rd'` fields of the `CIW`, `CL`, `CS`, `CA`, and `CB` formats.

RVC Register Number	000	001	010	011	100	101	110	111
Integer Register Number	x8	x9	x10	x11	x12	x13	x14	x15
Integer Register ABI Name	s0	s1	a0	a1	a2	a3	a4	a5
Floating-Point Register Number	f8	f9	f10	f11	f12	f13	f14	f15
Floating-Point Register ABI Name	fs0	fs1	fa0	fa1	fa2	fa3	fa4	fa5

Return Type	Bits<5>
Arguments	Bits<3> creg_idx

```
return {2'b01, creg_idx};
```

7.6. current_translation_mode

Returns the current first-stage translation mode for an explicit load or store from mode given the machine state (e.g., value of `satp` or `vsatp` csr).

Returns `SatpMode::Reserved` if the setting found in `satp` or `vsatp` is invalid.

Return Type	SatpMode
Arguments	PrivilegeMode mode

```
PrivilegeMode effective_mode = effective_ldst_mode();
if (effective_mode == PrivilegeMode::M) {
    return SatpMode::Bare;
}
if (CSR[misa].H == 1'b1) {
    if (effective_mode == PrivilegeMode::VS || effective_mode == PrivilegeMode::VU) {
        Bits<4> mode_val = CSR[vsatp].MODE;
        if (mode_val == $bits(SatpMode::Sv32)) {
            if (MXLEN == 64) {
                if ((effective_mode == PrivilegeMode::VS) && (CSR[hstatus].VSXL != $bits(XRegWidth::XLEN32))) {
                    return SatpMode::Reserved;
                }
                if ((effective_mode == PrivilegeMode::VU) && (CSR[vsstatus].UXL != $bits(XRegWidth::XLEN32))) {
                    return SatpMode::Reserved;
                }
            }
        }
    }
}
```

```

    if (!SV32_VSMODE_TRANSLATION) {
        return SatpMode::Reserved;
    }
    return SatpMode::Sv32;
} else if ((MXLEN == 64) && (mode_val == $bits(SatpMode::Sv39))) {
    if (effective_mode == PrivilegeMode::VS && CSR[hstatus].VSXL != $bits
(XRegWidth::XLEN64)) {
        return SatpMode::Reserved;
    }
    if (effective_mode == PrivilegeMode::VU && CSR[vsstatus].UXL != $bits
(XRegWidth::XLEN64)) {
        return SatpMode::Reserved;
    }
    if (!SV39_VSMODE_TRANSLATION) {
        return SatpMode::Reserved;
    }
    return SatpMode::Sv39;
} else if ((MXLEN == 64) && (mode_val == $bits(SatpMode::Sv48))) {
    if (effective_mode == PrivilegeMode::VS && CSR[hstatus].VSXL != $bits
(XRegWidth::XLEN64)) {
        return SatpMode::Reserved;
    }
    if (effective_mode == PrivilegeMode::VU && CSR[vsstatus].UXL != $bits
(XRegWidth::XLEN64)) {
        return SatpMode::Reserved;
    }
    if (!SV48_VSMODE_TRANSLATION) {
        return SatpMode::Reserved;
    }
    return SatpMode::Sv48;
} else if ((MXLEN == 64) && (mode_val == $bits(SatpMode::Sv57))) {
    if (effective_mode == PrivilegeMode::VS && CSR[hstatus].VSXL != $bits
(XRegWidth::XLEN64)) {
        return SatpMode::Reserved;
    }
    if (effective_mode == PrivilegeMode::VU && CSR[vsstatus].UXL != $bits
(XRegWidth::XLEN64)) {
        return SatpMode::Reserved;
    }
    if (!SV57_VSMODE_TRANSLATION) {
        return SatpMode::Reserved;
    }
    return SatpMode::Sv57;
} else {
    return SatpMode::Reserved;
}
}
} else if (CSR[misa].S == 1'b1) {
    assert(effective_mode == PrivilegeMode::S || effective_mode ==
PrivilegeMode::U, "unexpected priv mode");
    Bits<4> mode_val = CSR[satp].MODE;

```

```

if (mode_val == $bits(SatpMode::Sv32)) {
    if (MXLEN == 64) {
        if (effective_mode == PrivilegeMode::S && CSR[mstatus].SXL != $bits
(XRegWidth::XLEN32)) {
            return SatpMode::Reserved;
        }
        if (effective_mode == PrivilegeMode::U && CSR[sstatus].UXL != $bits
(XRegWidth::XLEN32)) {
            return SatpMode::Reserved;
        }
    }
    if (!implemented?(ExtensionName::Sv32)) {
        return SatpMode::Reserved;
    }
} else if ((MXLEN == 64) && (mode_val == $bits(SatpMode::Sv39))) {
    if (effective_mode == PrivilegeMode::S && CSR[mstatus].SXL != $bits
(XRegWidth::XLEN64)) {
        return SatpMode::Reserved;
    }
    if (effective_mode == PrivilegeMode::U && CSR[sstatus].UXL != $bits
(XRegWidth::XLEN64)) {
        return SatpMode::Reserved;
    }
    if (!implemented?(ExtensionName::Sv39)) {
        return SatpMode::Reserved;
    }
    return SatpMode::Sv39;
} else if ((MXLEN == 64) && (mode_val == $bits(SatpMode::Sv48))) {
    if (effective_mode == PrivilegeMode::S && CSR[mstatus].SXL != $bits
(XRegWidth::XLEN64)) {
        return SatpMode::Reserved;
    }
    if (effective_mode == PrivilegeMode::U && CSR[sstatus].UXL != $bits
(XRegWidth::XLEN64)) {
        return SatpMode::Reserved;
    }
    if (!implemented?(ExtensionName::Sv48)) {
        return SatpMode::Reserved;
    }
    return SatpMode::Sv48;
} else if ((MXLEN == 64) && (mode_val == $bits(SatpMode::Sv57))) {
    if (effective_mode == PrivilegeMode::S && CSR[mstatus].SXL != $bits
(XRegWidth::XLEN64)) {
        return SatpMode::Reserved;
    }
    if (effective_mode == PrivilegeMode::U && CSR[sstatus].UXL != $bits
(XRegWidth::XLEN64)) {
        return SatpMode::Reserved;
    }
    if (!implemented?(ExtensionName::Sv57)) {
        return SatpMode::Reserved;
    }
}

```

```
    }
    return SatpMode::Sv57;
} else {
    return SatpMode::Reserved;
}
}
```

7.7. delay (builtin)

Delay the processor by cycles cycles.

Return Type	void
Arguments	XReg cycles

7.8. effective_ldst_mode

Returns the effective privilege mode for normal explicit loads and stores, taking into account the current actual privilege mode and modifications from mstatus.MPRV.

Return Type	PrivilegeMode
Arguments	

```
if (mode() == PrivilegeMode::M) {
    if (CSR[misa].U == 1 && CSR[mstatus].MPRV == 1) {
        if (CSR[mstatus].MPP == 0b00) {
            if (CSR[misa].H == 1 && mpv() == 0b1) {
                return PrivilegeMode::VU;
            } else {
                return PrivilegeMode::U;
            }
        } else if (CSR[misa].S == 1 && CSR[mstatus].MPP == 0b01) {
            if (CSR[misa].H == 1 && mpv() == 0b1) {
                return PrivilegeMode::VS;
            } else {
                return PrivilegeMode::S;
            }
        }
    }
}
return mode();
```

7.9. exception_handling_mode

Returns the target privilege mode that will handle synchronous exception exception_code

Return Type	PrivilegeMode
Arguments	ExceptionCode exception_code

```

if (mode() == PrivilegeMode::M) {
    return PrivilegeMode::M;
} else if (implemented?(ExtensionName::S) && mode() == PrivilegeMode::HS) ||
(mode() == PrivilegeMode::U) {
    if (($bits(CSR[medeleg]) & (1 << $bits(exception_code))) != 0) {
        return PrivilegeMode::HS;
    } else {
        return PrivilegeMode::M;
    }
} else {
    assert(implemented?(ExtensionName::H) && mode() == PrivilegeMode::VS) ||
(mode() == PrivilegeMode::VU, "Unexpected mode");
    if (($bits(CSR[medeleg]) & (1 << $bits(exception_code))) != 0) {
        if (($bits(CSR[hdeleg]) & (1 << $bits(exception_code))) != 0) {
            return PrivilegeMode::VS;
        } else {
            return PrivilegeMode::HS;
        }
    } else {
        return PrivilegeMode::M;
    }
}

```

7.10. fence_tso (builtin)

Execute a TSO memory ordering fence.(according to the FENCE instruction).

Return Type	void
Arguments	

7.11. get_and_validate_stack_pointer

Validate and return stack pointer. Stack pointer is validated to fit in range, defined by qc.mstktopaddr and qc.mstkbottomaddr CSRs. In case when stack pointer is out-of-range, raising exception SpOutOfRange. Stack pointer is also validated to be 16-byte aligned. In case when stack pointer is not 16-bytes aligned, raising exception IllegalStackPointer.

Return Type	XReg
Arguments	XReg sp, Bits<INSTR_ENC_SIZE> encoding

```

if (sp < $bits(CSR[qc.mstkbottomaddr])) {
    raise(ExceptionCode::SpOutOfRange, mode(), encoding);
} else if (sp > $bits(CSR[qc.mstktopaddr])) {

```

```

    raise(ExceptionCode::SpOutOfRange, mode(), encoding);
} else if ((sp & 0xf) != 0) {
    raise(ExceptionCode::IllegalStackPointer, mode(), encoding);
}
return sp;

```

7.12. highest_set_bit

Returns the position of the highest (nearest MSB) bit that is '1', or -1 if value is zero.

Return Type	XReg
Arguments	XReg value

```

for (U32 i = xlen() - 1; i >= 0; i--) {
    if (value[i] == 1) {
        return i;
    }
}
return -'sd1;

```

7.13. ialign

Returns IALIGN, the smallest instruction encoding size, in bits.

Return Type	Bits<6>
Arguments	

```

if (implemented?(ExtensionName::C) && (CSR[misa].C == 0x1)) {
    return 16;
} else {
    return 32;
}

```

7.14. implemented? (generated)

Return true if the implementation supports extension.

Return Type	Boolean
Arguments	ExtensionName extension

7.15. implemented_version? (generated)

Return true if the implementation supports extension meeting 'version_requirement'.

Return Type	Boolean
Arguments	ExtensionName extension, String version_requirement

7.16. in_naturally_aligned_region?

Checks if a length-bit access starting at address lies entirely within an N-bit naturally-aligned region.

Return Type	Boolean
Arguments	XReg address, U32 length

```
XReg Mask = (N / 8) - 1;
return (address & ~Mask) == ((address + length - 1) & ~Mask);
```

7.17. is_naturally_aligned

Checks if value is naturally aligned to N bits.

Return Type	Boolean
Arguments	XReg value

```
return true if (N == 8);
XReg Mask = (N / 8) - 1;
return (value & ~Mask) == value;
```

7.18. iss_syscall (builtin)

Instruction set simulator system call.

Return Type	void
Arguments	XReg id, XReg arg

7.19. jump_halfword

Jump to virtual halfword address target_hw_addr.

If target address is misaligned, raise a MisalignedAddress exception.

Return Type	void
Arguments	XReg target_hw_addr

```
assert((target_hw_addr & 0x1) == 0x0, "Expected halfword-aligned address in
```

```

jump_halfword");
if (ialign() != 16) {
    if ((target_hw_addr & 0x3) != 0) {
        raise(ExceptionCode::InstructionAddressMisaligned, mode(), target_hw_addr);
    }
}
$pc = target_hw_addr;

```

7.20. lowest_set_bit

Returns the position of the lowest (nearest LSB) bit that is '1', or XLEN if value is zero.

Return Type	XReg
Arguments	XReg value

```

for (U32 i = 0; i < xlen(); i++) {
    if (value[i] == 1) {
        return i;
    }
}
return xlen();

```

7.21. maybe_cache_translation (generated)

Given a translation result, potentially cache the result for later use. This function models a TLB fill operation. A valid implementation does nothing.

Return Type	void
Arguments	XReg vaddr, MemoryOperation op, TranslationResult result

7.22. misaligned_is_atomic?

Returns true if an access starting at physical_address that is N bits long is atomic.

This function takes into account any Atomicity Granule PMAs, so **it should not be used for load-reserved/store-conditional**, since those PMAs do not apply to those accesses.

Return Type	Boolean
Arguments	Bits<PHYS_ADDR_WIDTH> physical_address

```

return false if (MISALIGNED_MAX_ATOMICITY_GRANULE_SIZE == 0);
if (pma_applies?(PmaAttribute::MAG16, physical_address, N) &&
    in_naturally_aligned_region?<128>(physical_address, N)) {
    return true;
}

```

```

} else if (pma_applies?(PmaAttribute::MAG8, physical_address, N) &&
in_naturally_aligned_region?(<4>(physical_address, N)) {
    return true;
} else if (pma_applies?(PmaAttribute::MAG4, physical_address, N) &&
in_naturally_aligned_region?(<32>(physical_address, N)) {
    return true;
} else if (pma_applies?(PmaAttribute::MAG2, physical_address, N) &&
in_naturally_aligned_region?(<16>(physical_address, N)) {
    return true;
} else {
    return false;
}

```

7.23. mode

Returns the current active privilege mode.

Return Type	PrivilegeMode
Arguments	

```

if (!implemented?(ExtensionName::S) && !implemented?(ExtensionName::U) &&
!implemented?(ExtensionName::H)) {
    return PrivilegeMode::M;
} else {
    return current_mode;
}

```

7.24. mtval_for

Given an exception code and a **legal** non-zero value for mtval, returns the value to be written in mtval considering implementation options

Return Type	XReg
Arguments	ExceptionCode exception_code, XReg tval

```

if (exception_code == ExceptionCode::Breakpoint) {
    return REPORT_VA_IN_MTVAL_ON_BREAKPOINT ? tval : 0;
} else if (exception_code == ExceptionCode::LoadAddressMisaligned) {
    return REPORT_VA_IN_MTVAL_ON_LOAD_MISALIGNED ? tval : 0;
} else if (exception_code == ExceptionCode::StoreAmoAddressMisaligned) {
    return REPORT_VA_IN_MTVAL_ON_STORE_AMO_MISALIGNED ? tval : 0;
} else if (exception_code == ExceptionCode::InstructionAddressMisaligned) {
    return REPORT_VA_IN_MTVAL_ON_INSTRUCTION_MISALIGNED ? tval : 0;
} else if (exception_code == ExceptionCode::LoadAccessFault) {
    return REPORT_VA_IN_MTVAL_ON_LOAD_ACCESS_FAULT ? tval : 0;
}

```

```

} else if (exception_code == ExceptionCode::StoreAmoAccessFault) {
    return REPORT_VA_IN_MTVAL_ON_STORE_AMO_ACCESS_FAULT ? tval : 0;
} else if (exception_code == ExceptionCode::InstructionAccessFault) {
    return REPORT_VA_IN_MTVAL_ON_INSTRUCTION_ACCESS_FAULT ? tval : 0;
} else if (exception_code == ExceptionCode::LoadPageFault) {
    return REPORT_VA_IN_MTVAL_ON_LOAD_PAGE_FAULT ? tval : 0;
} else if (exception_code == ExceptionCode::StoreAmoPageFault) {
    return REPORT_VA_IN_MTVAL_ON_STORE_AMO_PAGE_FAULT ? tval : 0;
} else if (exception_code == ExceptionCode::InstructionPageFault) {
    return REPORT_VA_IN_MTVAL_ON_INSTRUCTION_PAGE_FAULT ? tval : 0;
} else if (exception_code == ExceptionCode::IllegalInstruction) {
    return REPORT_ENCODING_IN_MTVAL_ON_ILLEGAL_INSTRUCTION ? tval : 0;
} else if (exception_code == ExceptionCode::SoftwareCheck) {
    return tval;
} else {
    return 0;
}

```

7.25. mtval_readonly?

Returns whether or not CSR[mtval] is read-only based on implementation options

Return Type	Boolean
Arguments	

```

return !(REPORT_VA_IN_MTVAL_ON_BREAKPOINT ||
REPORT_VA_IN_MTVAL_ON_LOAD_MISALIGNED ||
REPORT_VA_IN_MTVAL_ON_STORE_AMO_MISALIGNED ||
REPORT_VA_IN_MTVAL_ON_INSTRUCTION_MISALIGNED ||
REPORT_VA_IN_MTVAL_ON_LOAD_ACCESS_FAULT ||
REPORT_VA_IN_MTVAL_ON_STORE_AMO_ACCESS_FAULT ||
REPORT_VA_IN_MTVAL_ON_INSTRUCTION_ACCESS_FAULT ||
REPORT_VA_IN_MTVAL_ON_LOAD_PAGE_FAULT ||
REPORT_VA_IN_MTVAL_ON_STORE_AMO_PAGE_FAULT ||
REPORT_VA_IN_MTVAL_ON_INSTRUCTION_PAGE_FAULT ||
REPORT_ENCODING_IN_MTVAL_ON_ILLEGAL_INSTRUCTION ||
REPORT_CAUSE_IN_MTVAL_ON_SHADOW_STACK_SOFTWARE_CHECK ||
REPORT_CAUSE_IN_MTVAL_ON_LANDING_PAD_SOFTWARE_CHECK);

```

7.26. notify_mode_change (builtin)

Called whenever the privilege mode changes. Downstream tools can use this to hook events.

Return Type	void
Arguments	PrivilegeMode new_mode, PrivilegeMode old_mode

7.27. pma_applies? (builtin)

Checks if *attr* is applied to the entire physical address region between [*paddr*, *paddr* + *len*) based on static PMA attributes.

Return Type	Boolean
Arguments	PmaAttribute attr, Bits<PHYS_ADDR_WIDTH> paddr, U32 len

7.28. raise

Raise synchronous exception number *exception_code*.

The exception may be imprecise, and will cause execution to enter an unpredictable state, if `PRECISE_SYNCHRONOUS_EXCEPTIONS` is false.

Otherwise, the exception will be precise.

Return Type	void
Arguments	ExceptionCode exception_code, PrivilegeMode from_mode, XReg tval

```
if (!PRECISE_SYNCHRONOUS_EXCEPTIONS) {
    unpredictable("Imprecise synchronous exception");
} else {
    raise_precise(exception_code, from_mode, tval);
}
```

7.29. raise_precise

Raise synchronous exception number *exception_code*.

Return Type	void
Arguments	ExceptionCode exception_code, PrivilegeMode from_mode, XReg tval

```
PrivilegeMode handling_mode = exception_handling_mode(exception_code);
if (handling_mode == PrivilegeMode::M) {
    CSR[mepc].PC = $pc;
    if (!mtval_readonly?()) {
        CSR[mtval].VALUE = mtval_for(exception_code, tval);
    }
    $pc = {CSR[mtvec].BASE, 2'b00};
    CSR[mcause].INT = 1'b0;
    CSR[mcause].CODE = $bits(exception_code);
    if (CSR[misa].H == 1) {
```

```

CSR[mtval2].VALUE = 0;
CSR[mtinst].VALUE = 0;
if (from_mode == PrivilegeMode::VU || from_mode == PrivilegeMode::VS) {
    if (MXLEN == 32) {
        CSR[mstatush].MPV = 1;
    } else {
        CSR[mstatus].MPV = 1;
    }
} else {
    if (MXLEN == 32) {
        CSR[mstatush].MPV = 0;
    } else {
        CSR[mstatus].MPV = 0;
    }
}
}
CSR[mstatus].MPP = $bits(from_mode);
} else if (CSR[misa].S == 1 && (handling_mode == PrivilegeMode::S)) {
    CSR[sepc].PC = $pc;
    if (!stval_readonly?()) {
        CSR[stval].VALUE = stval_for(exception_code, tval);
    }
    $pc = {CSR[stvec].BASE, 2'b00};
    CSR[scause].INT = 1'b0;
    CSR[scause].CODE = $bits(exception_code);
    CSR[mstatus].SPP = $bits(from_mode)[0];
    if (CSR[misa].H == 1) {
        CSR[htval].VALUE = 0;
        CSR[htinst].VALUE = 0;
        CSR[hstatus].SPV = $bits(from_mode)[2];
        if (from_mode == PrivilegeMode::VU || from_mode == PrivilegeMode::VS) {
            CSR[hstatus].SPV = 1;
            if (exception_code == ExceptionCode::Breakpoint) && (
REPORT_VA_IN_STVAL_ON_BREAKPOINT || exception_code == ExceptionCode
::LoadAddressMisaligned) && (REPORT_VA_IN_STVAL_ON_LOAD_MISALIGNED ||
exception_code == ExceptionCode::StoreAmoAddressMisaligned) && (
REPORT_VA_IN_STVAL_ON_STORE_AMO_MISALIGNED || exception_code == ExceptionCode
::InstructionAddressMisaligned) && (
REPORT_VA_IN_STVAL_ON_INSTRUCTION_MISALIGNED || exception_code ==
ExceptionCode::LoadAccessFault) && (REPORT_VA_IN_STVAL_ON_LOAD_ACCESS_FAULT ||
exception_code == ExceptionCode::StoreAmoAccessFault) && (
REPORT_VA_IN_STVAL_ON_STORE_AMO_ACCESS_FAULT || exception_code ==
ExceptionCode::InstructionAccessFault) && (
REPORT_VA_IN_STVAL_ON_INSTRUCTION_ACCESS_FAULT || exception_code ==
ExceptionCode::LoadPageFault) && (REPORT_VA_IN_STVAL_ON_LOAD_PAGE_FAULT ||
exception_code == ExceptionCode::StoreAmoPageFault) && (
REPORT_VA_IN_STVAL_ON_STORE_AMO_PAGE_FAULT || exception_code == ExceptionCode
::InstructionPageFault) && (REPORT_VA_IN_STVAL_ON_INSTRUCTION_PAGE_FAULT) {
                CSR[hstatus].GVA = 1;
            } else {
                CSR[hstatus].GVA = 0;
            }
        }
    }
}

```

```

    }
    CSR[hstatus].SPVP = $bits(from_mode)[0];
  } else {
    CSR[hstatus].SPV = 0;
    CSR[hstatus].GVA = 0;
  }
}
} else if (CSR[misa].H == 1 && (handling_mode == PrivilegeMode::VS)) {
  CSR[vsepc].PC = $pc;
  if (!vstval_readonly?()) {
    CSR[vstval].VALUE = vstval_for(exception_code, tval);
  }
  $pc = {CSR[vstvec].BASE, 2'b00};
  CSR[vscause].INT = 1'b0;
  CSR[vscause].CODE = $bits(exception_code);
  CSR[vsstatus].SPP = $bits(from_mode)[0];
}
set_mode(handling_mode);
abort_current_instruction();

```

7.30. read_device_32 (builtin)

Read 32 bits from non-memory-mapped device. Such devices have own addresses not related to memory map.

Return Type	Bits<32>
Arguments	XReg dev_addr

7.31. read_memory

Read from virtual memory.

Return Type	Bits<LEN>
Arguments	XReg virtual_address, Bits<INSTR_ENC_SIZE> encoding

```

Boolean aligned = is_naturally_aligned<LEN>(virtual_address);
XReg physical_address;
if (aligned) {
  return read_memory_aligned<LEN>(virtual_address, encoding);
}
if (MISALIGNED_MAX_ATOMICITY_GRANULE_SIZE > 0) {
  assert(MISALIGNED_LDST_EXCEPTION_PRIORITY == "low", "Invalid config: can't
mix low-priority misaligned exceptions with large atomicity granule");
  physical_address = (CSR[misa].S == 1) ? translate(virtual_address,
MemoryOperation::Read, effective_ldst_mode(), encoding).paddr :
virtual_address;

```

```

    if (misaligned_is_atomic?<LEN>(physical_address)) {
        access_check(physical_address, LEN, virtual_address, MemoryOperation::Read,
ExceptionCode::LoadAccessFault, effective_ldst_mode());
        return read_physical_memory<LEN>(physical_address);
    }
}
if (!MISALIGNED_LDST) {
    if (MISALIGNED_LDST_EXCEPTION_PRIORITY == "low") {
        physical_address = (CSR[misa].S == 1) ? translate(virtual_address,
MemoryOperation::Read, effective_ldst_mode(), encoding).paddr :
virtual_address;
        access_check(physical_address, LEN, virtual_address, MemoryOperation::Read,
ExceptionCode::LoadAccessFault, effective_ldst_mode());
    }
    raise(ExceptionCode::LoadAddressMisaligned, effective_ldst_mode(),
virtual_address);
} else {
    if (MISALIGNED_SPLIT_STRATEGY == "by_byte") {
        Bits<LEN> result = 0;
        for (U32 i = 0; i <= (LEN / 8); i++) {
            result = result | (read_memory_aligned<8>(virtual_address + i, encoding)
<< (8 * i));
        }
        return result;
    } else if (MISALIGNED_SPLIT_STRATEGY == "custom") {
        unpredictable("An implementation is free to break a misaligned access any
way, leading to unpredictable behavior when any part of the misaligned access
causes an exception");
    }
}
}

```

7.32. read_memory_aligned

Read from virtual memory using a known aligned address.

Return Type	Bits<LEN>
Arguments	XReg virtual_address, Bits<INSTR_ENC_SIZE> encoding

```

TranslationResult result;
if (CSR[misa].S == 1) {
    result = translate(virtual_address, MemoryOperation::Read,
effective_ldst_mode(), encoding);
} else {
    result.paddr = virtual_address;
}
access_check(result.paddr, LEN, virtual_address, MemoryOperation::Read,
ExceptionCode::LoadAccessFault, effective_ldst_mode());

```

```
return read_physical_memory<LEN>(result.paddr);
```

7.33. read_physical_memory

Read from physical memory.

Return Type	Bits<len>
Arguments	XReg paddr

```
if (len == 8) {
    return read_physical_memory_8(paddr);
} else if (len == 16) {
    return read_physical_memory_16(paddr);
} else if (len == 32) {
    return read_physical_memory_32(paddr);
} else if (len == 64) {
    return read_physical_memory_64(paddr);
} else {
    assert(false, "Invalid len");
}
```

7.34. read_physical_memory_16 (builtin)

Read two bytes from physical memory.

Return Type	Bits<16>
Arguments	XReg paddr

7.35. read_physical_memory_32 (builtin)

Read four bytes from physical memory.

Return Type	Bits<32>
Arguments	XReg paddr

7.36. read_physical_memory_8 (builtin)

Read a byte from physical memory.

Return Type	Bits<8>
Arguments	XReg paddr

7.37. refresh_pending_interrupts

refreshes the calculation of a pending interrupt

needs to be called after any state update that could change a pending interrupt. This includes: - CSR[mip] - CSR[mie] - CSR[mstatus].MIE - CSR[mstatus].SIE - CSR[vsstatus].SIE - CSR[mideleg] - CSR[sideleg] - CSR[hideleg] - CSR[hvip] - CSR[hgeip] - CSR[hgeie] - mode changes

Return Type	void
Arguments	

```

Bits<MXLEN> pending_ints = CSR[mip].sw_read() & $bits(CSR[mie]);
if (pending_ints == 0) {
    pending_and_enabled_interrupts = 0;
    return ;
}
Boolean HAS_MIDELEG = implemented_version?(ExtensionName::S, "<= 1.9.1") ||
(implemented_version?(ExtensionName::S, "> 1.9.1") && implemented_version?
(ExtensionName::Sm, "> 1.9.1"));
Bits<MXLEN> mmode_enabled_ints = mode() == PrivilegeMode::M && (CSR[mstatus]
.MIE == 1'b0 ? 0 : ($bits(CSR[mie]) & (HAS_MIDELEG ? ~$bits(CSR[mideleg]) :
~MXLEN'0)));
Bits<MXLEN> mmode_pending_and_enabled = pending_ints & mmode_enabled_ints;
if (mmode_pending_and_enabled != 0) {
    pending_and_enabled_interrupts = mmode_pending_and_enabled;
    return ;
}
if (CSR[misa].S == 1'b1) {
    Bits<MXLEN> smode_enabled_ints = mode() == PrivilegeMode::M || (CSR[
mstatus].SIE == 1'b0 ? 0 : $bits(CSR[mie]) & ($bits(CSR[mideleg])));
    Bits<MXLEN> smode_pending_and_enabled = pending_ints & smode_enabled_ints;
    if (smode_pending_and_enabled != 0) {
        pending_and_enabled_interrupts = smode_pending_and_enabled;
        return ;
    }
}
pending_and_enabled_interrupts = 0;

```

7.38. set_mode

Set the current privilege mode to new_mode

Return Type	void
Arguments	PrivilegeMode new_mode

```

if (new_mode != current_mode) {
    notify_mode_change(new_mode, current_mode);
    current_mode = new_mode;
    refresh_pending_interrupts();
}

```

```
}

```

7.39. sext

Sign extend value starting at first_extended_bit.

Bits [XLEN-1: first_extended_bit] of the return value should get the value of bit (first_extended bit - 1).

Return Type	XReg
Arguments	XReg value, XReg first_extended_bit

```
if (first_extended_bit == MXLEN) {
    return value;
} else {
    Bits<1> sign = value[first_extended_bit - 1];
    for (U32 i = MXLEN - 1; i >= first_extended_bit; i--) {
        value[i] = sign;
    }
    return value;
}

```

7.40. sync_read_after_write_device (builtin)

Synchronize non-memory-mapped device read-after-write. Such devices have own addresses not related to memory map. Devices specified by device bitmask (up to 5 devices). Stalls processor till conditions met. Conditions are that last device output started or completed (depends on argument).

Return Type	void
Arguments	Boolean completed, Bits<5> device_bitmask

7.41. sync_write_after_read_device (builtin)

Synchronize non-memory-mapped device write-after-read. Such devices have own addresses not related to memory map. Devices specified by device bitmask (up to 5 devices). Stalls processor till conditions met. Conditions are that last device input started or completed (depends on argument).

Return Type	void
Arguments	Boolean completed, Bits<5> device_bitmask

7.42. translate

Translate a virtual address for operation type op that appears to execute at effective_mode.

The translation will depend on the effective privilege mode.

May raise a Page Fault or Access Fault.

The final physical address is **not** access checked (for PMP, PMA, etc., violations). (though intermediate page table reads will be)

Return Type	TranslationResult
Arguments	XReg vaddr, MemoryOperation op, PrivilegeMode effective_mode, Bits<INSTR_ENC_SIZE> encoding

```

Boolean cached_translation_valid;
CachedTranslationResult cached_translation_result;
cached_translation_result = cached_translation(vaddr, op);
if (cached_translation_result.valid) {
    return cached_translation_result.result;
}
TranslationResult result;
if (effective_mode == PrivilegeMode::M) {
    result.paddr = vaddr;
    return result;
}
SatpMode translation_mode = current_translation_mode(effective_mode);
if (translation_mode == SatpMode::Reserved) {
    if (op == MemoryOperation::Read) {
        raise(ExceptionCode::LoadPageFault, effective_mode, vaddr);
    } else if (op == MemoryOperation::Write || op == MemoryOperation
::ReadModifyWrite) {
        raise(ExceptionCode::StoreAmoPageFault, effective_mode, vaddr);
    } else {
        assert(op == MemoryOperation::Fetch, "Unexpected memory operation");
        raise(ExceptionCode::InstructionPageFault, effective_mode, vaddr);
    }
}
if (translation_mode == SatpMode::Sv32) {
    result = stage1_page_walk<32, 34, 32, 2>(vaddr, op, effective_mode,
encoding);
} else if (translation_mode == SatpMode::Sv39) {
    result = stage1_page_walk<39, 56, 64, 3>(vaddr, op, effective_mode,
encoding);
} else if (translation_mode == SatpMode::Sv48) {
    result = stage1_page_walk<48, 56, 64, 4>(vaddr, op, effective_mode,
encoding);
} else if (translation_mode == SatpMode::Sv57) {
    result = stage1_page_walk<57, 56, 64, 5>(vaddr, op, effective_mode,
encoding);
} else {
    assert(false, "Unexpected SatpMode");
}
maybe_cache_translation(vaddr, op, result);
return result;

```

7.43. unpredictable (builtin)

Indicate that the hart has reached a state that is unpredictable because the RISC-V spec allows multiple behaviors. Generally, this will be a fatal condition to any emulation, since it is unclear what to do next.

The single argument *why* is a string describing why the hart entered an unpredictable state.

Return Type	void
Arguments	String why

7.44. write_device_32 (builtin)

Write 32 bits to non-memory-mapped device. Such devices have own addresses not related to memory map.

Return Type	void
Arguments	XReg dev_addr, Bits<32> value

7.45. write_memory

Write to virtual memory

Return Type	void
Arguments	XReg virtual_address, Bits<LEN> value, Bits<INSTR_ENC_SIZE> encoding

```
Boolean aligned = is_naturally_aligned<LEN>(virtual_address);
XReg physical_address;
if (aligned) {
    write_memory_aligned<LEN>(virtual_address, value, encoding);
    return ;
}
if (MISALIGNED_MAX_ATOMICITY_GRANULE_SIZE > 0) {
    assert(MISALIGNED_LDST_EXCEPTION_PRIORITY == "low", "Invalid config: can't
mix low-priority misaligned exceptions with large atomicity granule");
    physical_address = (CSR[misa].S == 1) ? translate(virtual_address,
MemoryOperation::Write, effective_ldst_mode(), encoding).paddr :
virtual_address;
    if (misaligned_is_atomic?<LEN>(physical_address)) {
        access_check(physical_address, LEN, virtual_address, MemoryOperation::
Write, ExceptionCode::StoreAmoAccessFault, effective_ldst_mode());
        write_physical_memory<LEN>(physical_address, value);
        return ;
    }
}
```

```

if (!MISALIGNED_LDST) {
    if (MISALIGNED_LDST_EXCEPTION_PRIORITY == "low") {
        physical_address = (CSR[misa].S == 1) ? translate(virtual_address,
MemoryOperation::Write, effective_ldst_mode(), encoding).paddr :
virtual_address;
        access_check(physical_address, LEN, virtual_address, MemoryOperation::
Write, ExceptionCode::StoreAmoAccessFault, effective_ldst_mode());
    }
    raise(ExceptionCode::StoreAmoAddressMisaligned, effective_ldst_mode(),
virtual_address);
} else {
    if (MISALIGNED_SPLIT_STRATEGY == "by_byte") {
        for (U32 i = 0; i <= (LEN / 8); i++) {
            write_memory_aligned<8>(virtual_address + i, (value >> (8 * i))[7:0],
encoding);
        }
    } else if (MISALIGNED_SPLIT_STRATEGY == "custom") {
        unpredictable("An implementation is free to break a misaligned access any
way, leading to unpredictable behavior when any part of the misaligned access
causes an exception");
    }
}
}

```

7.46. write_memory_aligned

Write to virtual memory using a known aligned address.

Return Type	void
Arguments	XReg virtual_address, Bits<LEN> value, Bits<INSTR_ENC_SIZE> encoding

```

XReg physical_address;
physical_address = (CSR[misa].S == 1) ? translate(virtual_address,
MemoryOperation::Write, effective_ldst_mode(), encoding).paddr :
virtual_address;
access_check(physical_address, LEN, virtual_address, MemoryOperation::Write,
ExceptionCode::StoreAmoAccessFault, effective_ldst_mode());
write_physical_memory<LEN>(physical_address, value);

```

7.47. write_physical_memory

Write to physical memory.

Return Type	void
Arguments	XReg paddr, Bits<len> value

```

if (len == 8) {
    write_physical_memory_8(paddr, value);
} else if (len == 16) {
    write_physical_memory_16(paddr, value);
} else if (len == 32) {
    write_physical_memory_32(paddr, value);
} else if (len == 64) {
    write_physical_memory_64(paddr, value);
} else {
    assert(false, "Invalid len");
}

```

7.48. write_physical_memory_16 (builtin)

Write two bytes to physical memory.

Return Type	void
Arguments	XReg paddr, Bits<16> value

7.49. write_physical_memory_32 (builtin)

Write four bytes to physical memory.

Return Type	void
Arguments	XReg paddr, Bits<32> value

7.50. write_physical_memory_8 (builtin)

Write a byte to physical memory.

Return Type	void
Arguments	XReg paddr, Bits<8> value

7.51. xlen

Returns the effective XLEN for the current privilege mode.

Return Type	Bits<8>
Arguments	

```

if (MXLEN == 32) {
    return 32;
} else {
    if (mode() == PrivilegeMode::M) {

```

```

    if (CSR[misa].MXL == $bits(XRegWidth::XLEN32)) {
        return 32;
    } else if (CSR[misa].MXL == $bits(XRegWidth::XLEN64)) {
        return 64;
    }
} else if (implemented?(ExtensionName::S) && mode() == PrivilegeMode::S) {
    if (CSR[mstatus].SXL == $bits(XRegWidth::XLEN32)) {
        return 32;
    } else if (CSR[mstatus].SXL == $bits(XRegWidth::XLEN64)) {
        return 64;
    }
} else if (implemented?(ExtensionName::U) && mode() == PrivilegeMode::U) {
    if (CSR[mstatus].UXL == $bits(XRegWidth::XLEN32)) {
        return 32;
    } else if (CSR[mstatus].UXL == $bits(XRegWidth::XLEN64)) {
        return 64;
    }
} else if (implemented?(ExtensionName::H) && mode() == PrivilegeMode::VS) {
    if (CSR[hstatus].VSXL == $bits(XRegWidth::XLEN32)) {
        return 32;
    } else if (CSR[hstatus].VSXL == $bits(XRegWidth::XLEN64)) {
        return 64;
    }
} else if (implemented?(ExtensionName::H) && mode() == PrivilegeMode::VU) {
    if (CSR[vsstatus].UXL == $bits(XRegWidth::XLEN32)) {
        return 32;
    } else if (CSR[vsstatus].UXL == $bits(XRegWidth::XLEN64)) {
        return 64;
    }
}
}
}

```