# MC9S12XEP100
# Reference Manual
# Covers MC9S12XE Family

***HCS12X***
***Microcontrollers***

MC9S12XEP100RMV1

Rev. 1.25

02/2013

*freescale.com*

freescale

To provide the most up-to-date information, the document revision on the World Wide Web is the most current. A printed copy may be an earlier revision. To verif, refer to:

freescale.com

This document contains information for the complete S12XE-Family and thus includes a set of separate FTM module sections to cover the whole family. A full list of family members and options is included in the appendices.

This document contains information for all constituent modules, with the exception of the S12X CPU. For S12X CPU information please refer to CPU12XV2 in the CPU12/CPU12X Reference Manual.

## Revision History. Refer to module section revision history tables for more information.

| Date | Revision | Description |
|---|---|---|
| Sep, 2008 | 1.18 | Updated NVM timing parameter section for brownout case<br>Specified time delay from RESET to start of CPU code execution<br>Added NVM patch Part IDs<br>Enhanced ECT GPIO / timer function transitioning description |
| Dec, 2008 | 1.19 | Updated 208MAPBGA thermal parameters<br>Revised TIM flag clearing procedure<br>Corrected CRG register address<br>Added maskset identifier suffix for ATMC fab<br>Fixed typos |
| Aug, 2009 | 1.20 | Added 208MAPBGA disclaimer<br>Added VREAPI to PT5. Added LVR Note to electricals.<br>Updates to TIM/ECT/XGATE/SCI/MSCAN (see embedded rev. history) |
| Apr, 2010 | 1.21 | FTM section (see FTM revision history)<br>PIM section (see PIM revision history) |
| May, 2010 | 1.22 | ECT and TIM sections (see ECT, TIM revision history tables)<br>BDM Alternate clock source defined in device overview |
| Sep, 2010 | 1.23 | Added S12XEG256 option. Updated MSCAN section |
| Aug, 2012 | 1.24 | Added bandgap voltage to electricals<br>Added new maskset and Part ID numbers<br>Minor updates to MSCAN,SCI and S12XINT sections<br>Removed BGA disclaimer |
| Feb, 2013 | 1.25 | Updated MSCAN section<br>Formatting updates and minor corrections in PWM, CRG, BDM, DBG sections<br>Updated Ordering Information |

Chapter 1 Device Overview MC9S12XE-Family . . . . . 27
Chapter 2 Port Integration Module (S12XEP100PIMV1) . . . . . 89
Chapter 3 Memory Mapping Control (S12XMMCV4) . . . . . 187
Chapter 4 Memory Protection Unit (S12XMPUV1) . . . . . 227
Chapter 5 External Bus Interface (S12XEBIV4) . . . . . 241
Chapter 6 Interrupt (S12XINTV2) . . . . . 261
Chapter 7 Background Debug Module (S12XBDMV2) . . . . . 279
Chapter 8 S12X Debug (S12XDBGV3) Module . . . . . 305
Chapter 9 Security (S12XE9SECV2) . . . . . 347
Chapter 10 XGATE (S12XGATEV3) . . . . . 353
Chapter 11 S12XE Clocks and Reset Generator (S12XECRGV1) . . . . . 469
Chapter 12 Pierce Oscillator (S12XOSCLCPV2) . . . . . 499
Chapter 13 Analog-to-Digital Converter (ADC12B16CV1) . . . . . 503
Chapter 14 Enhanced Capture Timer (ECT16B8CV3) . . . . . 527
Chapter 15 Inter-Integrated Circuit (IICV3) Block Description . . . . . 579
Chapter 16 Scalable Controller Area Network (S12MSCANV3) . . . . . 605
Chapter 17 Periodic Interrupt Timer (S12PIT24B8CV2) . . . . . 659
Chapter 18 Periodic Interrupt Timer (S12PIT24B4CV2) . . . . . 677
Chapter 19 Pulse-Width Modulator (S12PWM8B8CV1) . . . . . 691
Chapter 20 Serial Communication Interface (S12SCIV5) . . . . . 723
Chapter 21 Serial Peripheral Interface (S12SPIV5) . . . . . 761
Chapter 22 Timer Module (TIM16B8CV2) Block Description . . . . . 787
Chapter 23 Voltage Regulator (S12VREGL3V3V1) . . . . . 815
Chapter 24 128 KByte Flash Module (S12XFTM128K2V1) . . . . . 832
Chapter 25 256 KByte Flash Module (S12XFTM256K2V1) . . . . . 891

Chapter 26 384 KByte Flash Module (S12XFTM384K2V1) . . . . . .953
Chapter 27 512 KByte Flash Module (S12XFTM512K3V1) . . . . . .1016
Chapter 28 768 KByte Flash Module (S12XFTM768K4V2) . . . . . .1077
Chapter 29 1024 KByte Flash Module (S12XFTM1024K5V2) . . . . . .1140
Appendix A Electrical Characteristics. . . . . .1201
Appendix B Package Information . . . . . .1258
Appendix C PCB Layout Guidelines . . . . . .1260
Appendix D Derivative Differences . . . . . .1268
Appendix E Detailed Register Address Map. . . . . .1271
Appendix F Ordering Information . . . . . .1322

# Blank Page

# Blank Page

# Chapter 1
# Device Overview MC9S12XE-Family


1.1 Introduction . . . 27
1.1.1 Features . . . 27
1.1.2 Modes of Operation . . . 31
1.1.3 Block Diagram . . . 32
1.1.4 Device Memory Map . . . 33
1.1.5 Address Mapping . . . 34
1.1.6 Detailed Register Map . . . 40
1.1.7 Part ID Assignments . . . 40
1.2 Signal Description . . . 41
1.2.1 Device Pinout . . . 42
1.2.2 Pin Assignment Overview . . . 48
1.2.3 Detailed Signal Descriptions . . . 62
1.2.4 Power Supply Pins . . . 72
1.3 System Clock Description . . . 75
1.4 Modes of Operation . . . 76
1.4.1 Chip Configuration Summary . . . 76
1.4.2 Power Modes . . . 78
1.4.3 Freeze Mode . . . 79
1.4.4 System States . . . 79
1.5 Security . . . 80
1.6 Resets and Interrupts . . . 80
1.6.1 Resets . . . 80
1.6.2 Vectors . . . 80
1.6.3 Effects of Reset . . . 84
1.7 ADC0 Configuration . . . 86
1.7.1 External Trigger Input Connection . . . 86
1.7.2 ADC0 Channel[17] Connection . . . 86
1.8 ADC1 External Trigger Input Connection . . . 86
1.9 MPU Configuration . . . 87
1.10 VREG Configuration . . . 87
1.10.1 Temperature Sensor Configuration . . . 87
1.11 BDM Clock Configuration . . . 87
1.12 S12XEPIM Configuration . . . 87
1.13 Oscillator Configuration . . . 88


# Chapter 2
# Port Integration Module (S12XEPIMV1)


2.1 Introduction . . . 89

2.1.1 Overview . . . 89
2.1.2 Features . . . 90
2.2 External Signal Description . . . 90
2.3 Memory Map and Register Definition . . . 98
2.3.1 Memory Map . . . 99
2.3.2 Register Descriptions . . . 106
2.3.3 Port A Data Register (PORTA) . . . 108
2.3.4 Port B Data Register (PORTB) . . . 108
2.3.5 Port A Data Direction Register (DDRA) . . . 109
2.3.6 Port B Data Direction Register (DDRB) . . . 109
2.3.7 Port C Data Register (PORTC) . . . 110
2.3.8 Port D Data Register (PORTD) . . . 111
2.3.9 Port C Data Direction Register (DDRC) . . . 111
2.3.10 Port D Data Direction Register (DDRD) . . . 112
2.3.11 Port E Data Register (PORTE) . . . 113
2.3.12 Port E Data Direction Register (DDRE) . . . 114
2.3.13 S12X_EBI ports, BKGD pin Pull-up Control Register (PUCR) . . . 114
2.3.14 S12X_EBI ports Reduced Drive Register (RDRIV) . . . 116
2.3.15 ECLK Control Register (ECLKCTL) . . . 117
2.3.16 PIM Reserved Register . . . 118
2.3.17 IRQ Control Register (IRQCR) . . . 119
2.3.18 PIM Reserved Register . . . 119
2.3.19 Port K Data Register (PORTK) . . . 120
2.3.20 Port K Data Direction Register (DDRK) . . . 120
2.3.21 Port T Data Register (PTT) . . . 121
2.3.22 Port T Input Register (PTIT) . . . 122
2.3.23 Port T Data Direction Register (DDRT) . . . 122
2.3.24 Port T Reduced Drive Register (RDRT) . . . 123
2.3.25 Port T Pull Device Enable Register (PERT) . . . 123
2.3.26 Port T Polarity Select Register (PPST) . . . 124
2.3.27 PIM Reserved Register . . . 124
2.3.28 PIM Reserved Register . . . 124
2.3.29 Port S Data Register (PTS) . . . 125
2.3.30 Port S Input Register (PTIS) . . . 126
2.3.31 Port S Data Direction Register (DDRS) . . . 126
2.3.32 Port S Reduced Drive Register (RDRS) . . . 127
2.3.33 Port S Pull Device Enable Register (PERS) . . . 128
2.3.34 Port S Polarity Select Register (PPSS) . . . 128
2.3.35 Port S Wired-Or Mode Register (WOMS) . . . 129
2.3.36 PIM Reserved Register . . . 129
2.3.37 Port M Data Register (PTM) . . . 130
2.3.38 Port M Input Register (PTIM) . . . 131
2.3.39 Port M Data Direction Register (DDRM) . . . 132
2.3.40 Port M Reduced Drive Register (RDRM) . . . 134
2.3.41 Port M Pull Device Enable Register (PERM) . . . 134

2.3.42 Port M Polarity Select Register (PPSM) . . . 135
2.3.43 Port M Wired-Or Mode Register (WOMM) . . . 135
2.3.44 Module Routing Register (MODRR) . . . 136
2.3.45 Port P Data Register (PTP) . . . 137
2.3.46 Port P Input Register (PTIP) . . . 138
2.3.47 Port P Data Direction Register (DDRP) . . . 139
2.3.48 Port P Reduced Drive Register (RDRP) . . . 140
2.3.49 Port P Pull Device Enable Register (PERP) . . . 140
2.3.50 Port P Polarity Select Register (PPSP) . . . 141
2.3.51 Port P Interrupt Enable Register (PIEP) . . . 141
2.3.52 Port P Interrupt Flag Register (PIFP) . . . 142
2.3.53 Port H Data Register (PTH) . . . 142
2.3.54 Port H Input Register (PTIH) . . . 144
2.3.55 Port H Data Direction Register (DDRH) . . . 144
2.3.56 Port H Reduced Drive Register (RDRH) . . . 146
2.3.57 Port H Pull Device Enable Register (PERH) . . . 147
2.3.58 Port H Polarity Select Register (PPSH) . . . 147
2.3.59 Port H Interrupt Enable Register (PIEH) . . . 148
2.3.60 Port H Interrupt Flag Register (PIFH) . . . 148
2.3.61 Port J Data Register (PTJ) . . . 149
2.3.62 Port J Input Register (PTIJ) . . . 150
2.3.63 Port J Data Direction Register (DDRJ) . . . 150
2.3.64 Port J Reduced Drive Register (RDRJ) . . . 152
2.3.65 Port J Pull Device Enable Register (PERJ) . . . 153
2.3.66 Port J Polarity Select Register (PPSJ) . . . 153
2.3.67 Port J Interrupt Enable Register (PIEJ) . . . 154
2.3.68 Port J Interrupt Flag Register (PIFJ) . . . 154
2.3.69 Port AD0 Data Register 0 (PT0AD0) . . . 155
2.3.70 Port AD0 Data Register 1 (PT1AD0) . . . 155
2.3.71 Port AD0 Data Direction Register 0 (DDR0AD0) . . . 156
2.3.72 Port AD0 Data Direction Register 1 (DDR1AD0) . . . 156
2.3.73 Port AD0 Reduced Drive Register 0 (RDR0AD0) . . . 157
2.3.74 Port AD0 Reduced Drive Register 1 (RDR1AD0) . . . 158
2.3.75 Port AD0 Pull Up Enable Register 0 (PER0AD0) . . . 158
2.3.76 Port AD0 Pull Up Enable Register 1 (PER1AD0) . . . 159
2.3.77 Port AD1 Data Register 0 (PT0AD1) . . . 159
2.3.78 Port AD1 Data Register 1 (PT1AD1) . . . 160
2.3.79 Port AD1 Data Direction Register 0 (DDR0AD1) . . . 160
2.3.80 Port AD1 Data Direction Register 1 (DDR1AD1) . . . 161
2.3.81 Port AD1 Reduced Drive Register 0 (RDR0AD1) . . . 162
2.3.82 Port AD1 Reduced Drive Register 1 (RDR1AD1) . . . 162
2.3.83 Port AD1 Pull Up Enable Register 0 (PER0AD1) . . . 163
2.3.84 Port AD1 Pull Up Enable Register 1 (PER1AD1) . . . 163
2.3.85 Port R Data Register (PTR) . . . 164
2.3.86 Port R Input Register (PTIR) . . . 164

2.3.87 Port R Data Direction Register (DDRR) . . . 165
2.3.88 Port R Reduced Drive Register (RDRR) . . . 165
2.3.89 Port R Pull Device Enable Register (PERR) . . . 166
2.3.90 Port R Polarity Select Register (PPSR) . . . 166
2.3.91 PIM Reserved Register . . . 167
2.3.92 Port R Routing Register (PTRRR) . . . 167
2.3.93 Port L Data Register (PTL) . . . 168
2.3.94 Port L Input Register (PTIL) . . . 170
2.3.95 Port L Data Direction Register (DDRL) . . . 170
2.3.96 Port L Reduced Drive Register (RDRL) . . . 171
2.3.97 Port L Pull Device Enable Register (PERL) . . . 171
2.3.98 Port L Polarity Select Register (PPSL) . . . 172
2.3.99 Port L Wired-Or Mode Register (WOML) . . . 172
2.3.100 Port L Routing Register (PTLRR) . . . 173
2.3.101 Port F Data Register (PTF) . . . 173
2.3.102 Port F Input Register (PTIF) . . . 175
2.3.103 Port F Data Direction Register (DDRF) . . . 175
2.3.104 Port F Reduced Drive Register (RDRF) . . . 176
2.3.105 Port F Pull Device Enable Register (PERF) . . . 176
2.3.106 Port F Polarity Select Register (PPSF) . . . 177
2.3.107 PIM Reserved Register . . . 177
2.3.108 Port F Routing Register (PTFRR) . . . 177
2.4 Functional Description . . . 178
2.4.1 General . . . 178
2.4.2 Registers . . . 178
2.4.3 Pins and Ports . . . 181
2.4.4 Pin interrupts . . . 185
2.5 Initialization Information . . . 186
2.5.1 Port Data and Data Direction Register writes . . . 186

## Chapter 3
## Memory Mapping Control (S12XMMCV4)

3.1 Introduction . . . 187
3.1.1 Terminology . . . 188
3.1.2 Features . . . 188
3.1.3 S12X Memory Mapping . . . 189
3.1.4 Modes of Operation . . . 189
3.1.5 Block Diagram . . . 190
3.2 External Signal Description . . . 190
3.3 Memory Map and Registers . . . 192
3.3.1 Module Memory Map . . . 192
3.3.2 Register Descriptions . . . 193
3.4 Functional Description . . . 204
3.4.1 MCU Operating Mode . . . 204
3.4.2 Memory Map Scheme . . . 205

3.4.3 Chip Access Restrictions . . . 217
3.4.4 Chip Bus Control . . . 217
3.5 Initialization/Application Information . . . 218
3.5.1 CALL and RTC Instructions . . . 218
3.5.2 Port Replacement Registers (PRRs) . . . 220
3.5.3 On-Chip ROM Control . . . 221

## Chapter 4
## Memory Protection Unit (S12XMPUV1)

4.1 Introduction . . . 227
4.1.1 Preface . . . 227
4.1.2 Overview . . . 227
4.1.3 Features . . . 228
4.1.4 Modes of Operation . . . 229
4.2 External Signal Description . . . 229
4.3 Memory Map and Register Definition . . . 229
4.3.1 Register Descriptions . . . 230
4.4 Functional Description . . . 237
4.4.1 Protection Descriptors . . . 237
4.4.2 Interrupts . . . 239
4.5 Initialization/Application Information . . . 239
4.5.1 Initialization . . . 239

## Chapter 5
## External Bus Interface (S12XEBIV4)

5.1 Introduction . . . 241
5.1.1 Glossary or Terms . . . 242
5.1.2 Features . . . 242
5.1.3 Modes of Operation . . . 242
5.1.4 Block Diagram . . . 243
5.2 External Signal Description . . . 243
5.3 Memory Map and Register Definition . . . 245
5.3.1 Module Memory Map . . . 245
5.3.2 Register Descriptions . . . 245
5.4 Functional Description . . . 248
5.4.1 Operating Modes and External Bus Properties . . . 248
5.4.2 Internal Visibility . . . 249
5.4.3 Accesses to Port Replacement Registers . . . 253
5.4.4 Stretched External Bus Accesses . . . 253
5.4.5 Data Select and Data Direction Signals . . . 254
5.4.6 Low-Power Options . . . 256
5.5 Initialization/Application Information . . . 256
5.5.1 Normal Expanded Mode . . . 256
5.5.2 Emulation Modes . . . 257

## Chapter 6
## Interrupt (S12XINTV2)


6.1 Introduction . . . 261
6.1.1 Glossary . . . 262
6.1.2 Features . . . 262
6.1.3 Modes of Operation . . . 263
6.1.4 Block Diagram . . . 264
6.2 External Signal Description . . . 264
6.3 Memory Map and Register Definition . . . 265
6.3.1 Module Memory Map . . . 265
6.3.2 Register Descriptions . . . 266
6.4 Functional Description . . . 271
6.4.1 S12X Exception Requests . . . 272
6.4.2 Interrupt Prioritization . . . 272
6.4.3 XGATE Requests . . . 273
6.4.4 Priority Decoders . . . 273
6.4.5 Reset Exception Requests . . . 274
6.4.6 Exception Priority . . . 274
6.5 Initialization/Application Information . . . 275
6.5.1 Initialization . . . 275
6.5.2 Interrupt Nesting . . . 275
6.5.3 Wake Up from Stop or Wait Mode . . . 276


## Chapter 7
## Background Debug Module (S12XBDMV2)


7.1 Introduction . . . 279
7.1.1 Features . . . 279
7.1.2 Modes of Operation . . . 280
7.1.3 Block Diagram . . . 281
7.2 External Signal Description . . . 281
7.3 Memory Map and Register Definition . . . 282
7.3.1 Module Memory Map . . . 282
7.3.2 Register Descriptions . . . 282
7.3.3 Family ID Assignment . . . 287
7.4 Functional Description . . . 287
7.4.1 Security . . . 288
7.4.2 Enabling and Activating BDM . . . 288
7.4.3 BDM Hardware Commands . . . 289
7.4.4 Standard BDM Firmware Commands . . . 290
7.4.5 BDM Command Structure . . . 291
7.4.6 BDM Serial Interface . . . 293
7.4.7 Serial Interface Hardware Handshake Protocol . . . 296
7.4.8 Hardware Handshake Abort Procedure . . . 298
7.4.9 SYNC — Request Timed Reference Pulse . . . 301

7.4.10 Instruction Tracing . . . 302
7.4.11 Serial Communication Time Out . . . 303

## Chapter 8
## S12X Debug (S12XDBGV3) Module

8.1 Introduction . . . 305
8.1.1 Glossary . . . 305
8.1.2 Overview . . . 306
8.1.3 Features . . . 306
8.1.4 Modes of Operation . . . 307
8.1.5 Block Diagram . . . 308
8.2 External Signal Description . . . 308
8.3 Memory Map and Registers . . . 308
8.3.1 Module Memory Map . . . 308
8.3.2 Register Descriptions . . . 310
8.4 Functional Description . . . 326
8.4.1 S12XDBG Operation . . . 327
8.4.2 Comparator Modes . . . 327
8.4.3 Trigger Modes . . . 331
8.4.4 State Sequence Control . . . 332
8.4.5 Trace Buffer Operation . . . 333
8.4.6 Tagging . . . 341
8.4.7 Breakpoints . . . 342

## Chapter 9
## Security (S12XE9SECV2)

9.1 Introduction . . . 347
9.1.1 Features . . . 347
9.1.2 Modes of Operation . . . 348
9.1.3 Securing the Microcontroller . . . 348
9.1.4 Operation of the Secured Microcontroller . . . 349
9.1.5 Unsecuring the Microcontroller . . . 350
9.1.6 Reprogramming the Security Bits . . . 351
9.1.7 Complete Memory Erase (Special Modes) . . . 351

## Chapter 10
## XGATE (S12XGATEV3)

10.1 Introduction . . . 353
10.1.1 Glossary of Terms . . . 353
10.1.2 Features . . . 354
10.1.3 Modes of Operation . . . 355
10.1.4 Block Diagram . . . 355
10.2 External Signal Description . . . 356
10.3 Memory Map and Register Definition . . . 356

10.3.1 Register Descriptions . . . 356
10.4 Functional Description . . . 373
10.4.1 XGATE RISC Core . . . 374
10.4.2 Programmer's Model . . . 374
10.4.3 Memory Map . . . 375
10.4.4 Semaphores . . . 376
10.4.5 Software Error Detection . . . 378
10.5 Interrupts . . . 379
10.5.1 Incoming Interrupt Requests . . . 379
10.5.2 Outgoing Interrupt Requests . . . 379
10.6 Debug Mode . . . 379
10.6.1 Debug Features . . . 379
10.6.2 Leaving Debug Mode . . . 381
10.7 Security . . . 381
10.8 Instruction Set . . . 382
10.8.1 Addressing Modes . . . 382
10.8.2 Instruction Summary and Usage . . . 385
10.8.3 Cycle Notation . . . 387
10.8.4 Thread Execution . . . 388
10.8.5 Instruction Glossary . . . 388
10.8.6 Instruction Coding . . . 461
10.9 Initialization and Application Information . . . 463
10.9.1 Initialization . . . 463
10.9.2 Code Example (Transmit "Hello World!" on SCI) . . . 463
10.9.3 Stack Support . . . 466

## Chapter 11
## S12XE Clocks and Reset Generator (S12XECRGV1)

11.1 Introduction . . . 469
11.1.1 Features . . . 469
11.1.2 Modes of Operation . . . 470
11.1.3 Block Diagram . . . 470
11.2 Signal Description . . . 471
11.2.1 $V_{DDPLL}$, $V_{SSPLL}$ . . . 471
11.2.2 $\overline{RESET}$ . . . 471
11.3 Memory Map and Registers . . . 472
11.3.1 Module Memory Map . . . 472
11.3.2 Register Descriptions . . . 473
11.4 Functional Description . . . 486
11.4.1 Functional Blocks . . . 486
11.4.2 Operation Modes . . . 491
11.4.3 Low Power Options . . . 492
11.5 Resets . . . 494
11.5.1 Description of Reset Operation . . . 495
11.6 Interrupts . . . 497

11.6.1 Description of Interrupt Operation . . . . . 498

## Chapter 12
## Pierce Oscillator (S12XOSCLCPV2)

12.1 Introduction . . . . . 499
12.1.1 Features . . . . . 499
12.1.2 Modes of Operation . . . . . 499
12.1.3 Block Diagram . . . . . 500
12.2 External Signal Description . . . . . 500
12.2.1 $V_{DDPLL}$ and $V_{SSPLL}$ — Operating and Ground Voltage Pins . . . . . 500
12.2.2 EXTAL and XTAL — Input and Output Pins . . . . . 500
12.3 Memory Map and Register Definition . . . . . 502
12.4 Functional Description . . . . . 502
12.4.1 Gain Control . . . . . 502
12.4.2 Clock Monitor . . . . . 502
12.4.3 Wait Mode Operation . . . . . 502
12.4.4 Stop Mode Operation . . . . . 502

## Chapter 13
## Analog-to-Digital Converter (ADC12B16CV1)

13.1 Introduction . . . . . 503
13.1.1 Features . . . . . 503
13.1.2 Modes of Operation . . . . . 504
13.1.3 Block Diagram . . . . . 505
13.2 Signal Description . . . . . 506
13.2.1 Detailed Signal Descriptions . . . . . 506
13.3 Memory Map and Register Definition . . . . . 506
13.3.1 Module Memory Map . . . . . 506
13.3.2 Register Descriptions . . . . . 508
13.4 Functional Description . . . . . 523
13.4.1 Analog Sub-Block . . . . . 523
13.4.2 Digital Sub-Block . . . . . 524
13.5 Resets . . . . . 525
13.6 Interrupts . . . . . 525

## Chapter 14
## Enhanced Capture Timer (ECT16B8CV3)

14.1 Introduction . . . . . 527
14.1.1 Features . . . . . 527
14.1.2 Modes of Operation . . . . . 528
14.1.3 Block Diagram . . . . . 529
14.2 External Signal Description . . . . . 529
14.2.1 IOC7 — Input Capture and Output Compare Channel 7 . . . . . 529
14.2.2 IOC6 — Input Capture and Output Compare Channel 6 . . . . . 529

14.2.3 IOC5 — Input Capture and Output Compare Channel 5 . . . . . 530
14.2.4 IOC4 — Input Capture and Output Compare Channel 4 . . . . . 530
14.2.5 IOC3 — Input Capture and Output Compare Channel 3 . . . . . 530
14.2.6 IOC2 — Input Capture and Output Compare Channel 2 . . . . . 530
14.2.7 IOC1 — Input Capture and Output Compare Channel 1 . . . . . 530
14.2.8 IOC0 — Input Capture and Output Compare Channel 0 . . . . . 530
14.3 Memory Map and Register Definition . . . . . 530
14.3.1 Module Memory Map . . . . . 530
14.3.2 Register Descriptions . . . . . 530
14.4 Functional Description . . . . . 566
14.4.1 Enhanced Capture Timer Modes of Operation . . . . . 573
14.4.2 Reset . . . . . 577
14.4.3 Interrupts . . . . . 577

## Chapter 15
## Inter-Integrated Circuit (IICV3) Block Description

15.1 Introduction . . . . . 579
15.1.1 Features . . . . . 579
15.1.2 Modes of Operation . . . . . 580
15.1.3 Block Diagram . . . . . 580
15.2 External Signal Description . . . . . 580
15.2.1 IIC_SCL — Serial Clock Line Pin . . . . . 580
15.2.2 IIC_SDA — Serial Data Line Pin . . . . . 580
15.3 Memory Map and Register Definition . . . . . 581
15.3.1 Register Descriptions . . . . . 581
15.4 Functional Description . . . . . 593
15.4.1 I-Bus Protocol . . . . . 593
15.4.2 Operation in Run Mode . . . . . 598
15.4.3 Operation in Wait Mode . . . . . 598
15.4.4 Operation in Stop Mode . . . . . 598
15.5 Resets . . . . . 598
15.6 Interrupts . . . . . 598
15.7 Application Information . . . . . 599
15.7.1 IIC Programming Examples . . . . . 599

## Chapter 16
## Freescale’s Scalable Controller Area Network (S12MSCANV3)

16.1 Introduction . . . . . 605
16.1.1 Glossary . . . . . 606
16.1.2 Block Diagram . . . . . 606
16.1.3 Features . . . . . 607
16.1.4 Modes of Operation . . . . . 607
16.2 External Signal Description . . . . . 608
16.2.1 RXCAN — CAN Receiver Input Pin . . . . . 608

16.2.2 TXCAN — CAN Transmitter Output Pin . . . 608
16.2.3 CAN System . . . 608
16.3 Memory Map and Register Definition . . . 609
16.3.1 Module Memory Map . . . 609
16.3.2 Register Descriptions . . . 611
16.3.3 Programmer's Model of Message Storage . . . 630
16.4 Functional Description . . . 641
16.4.1 General . . . 641
16.4.2 Message Storage . . . 641
16.4.3 Identifier Acceptance Filter . . . 644
16.4.4 Modes of Operation . . . 650
16.4.5 Low-Power Options . . . 652
16.4.6 Reset Initialization . . . 656
16.4.7 Interrupts . . . 656
16.5 Initialization/Application Information . . . 658
16.5.1 MSCAN initialization . . . 658
16.5.2 Bus-Off Recovery . . . 658

## Chapter 17
## Periodic Interrupt Timer (S12PIT24B8CV2)

17.1 Introduction . . . 659
17.1.1 Glossary . . . 659
17.1.2 Features . . . 659
17.1.3 Modes of Operation . . . 659
17.1.4 Block Diagram . . . 660
17.2 External Signal Description . . . 660
17.3 Register Definition . . . 661
17.4 Functional Description . . . 671
17.4.1 Timer . . . 672
17.4.2 Interrupt Interface . . . 673
17.4.3 Hardware Trigger . . . 673
17.5 Initialization . . . 674
17.5.1 Startup . . . 674
17.5.2 Shutdown . . . 674
17.5.3 Flag Clearing . . . 674
17.6 Application Information . . . 674

## Chapter 18
## Periodic Interrupt Timer (S12PIT24B4CV2)

18.1 Introduction . . . 677
18.1.1 Glossary . . . 677
18.1.2 Features . . . 677
18.1.3 Modes of Operation . . . 677
18.1.4 Block Diagram . . . 678

18.2 External Signal Description . . . . . 678
18.3 Register Definition . . . . . 678
18.4 Functional Description . . . . . 687
18.4.1 Timer . . . . . 687
18.4.2 Interrupt Interface . . . . . 689
18.4.3 Hardware Trigger . . . . . 689
18.5 Initialization . . . . . 689
18.5.1 Startup . . . . . 689
18.5.2 Shutdown . . . . . 689
18.5.3 Flag Clearing . . . . . 689
18.6 Application Information . . . . . 690

## Chapter 19
## Pulse-Width Modulator (S12PWM8B8CV1)

19.1 Introduction . . . . . 691
19.1.1 Features . . . . . 691
19.1.2 Modes of Operation . . . . . 691
19.1.3 Block Diagram . . . . . 692
19.2 External Signal Description . . . . . 692
19.2.1 PWM7 — PWM Channel 7 . . . . . 692
19.2.2 PWM6 — PWM Channel 6 . . . . . 693
19.2.3 PWM5 — PWM Channel 5 . . . . . 693
19.2.4 PWM4 — PWM Channel 4 . . . . . 693
19.2.5 PWM3 — PWM Channel 3 . . . . . 693
19.2.6 PWM3 — PWM Channel 2 . . . . . 693
19.2.7 PWM3 — PWM Channel 1 . . . . . 693
19.2.8 PWM3 — PWM Channel 0 . . . . . 693
19.3 Memory Map and Register Definition . . . . . 693
19.3.1 Module Memory Map . . . . . 693
19.3.2 Register Descriptions . . . . . 694
19.4 Functional Description . . . . . 709
19.4.1 PWM Clock Select . . . . . 709
19.4.2 PWM Channel Timers . . . . . 712
19.5 Resets . . . . . 720
19.6 Interrupts . . . . . 721

## Chapter 20
## Serial Communication Interface (S12SCIV5)

20.1 Introduction . . . . . 723
20.1.1 Glossary . . . . . 723
20.1.2 Features . . . . . 724
20.1.3 Modes of Operation . . . . . 724
20.1.4 Block Diagram . . . . . 725
20.2 External Signal Description . . . . . 726

20.2.1 TXD — Transmit Pin . . . 726
20.2.2 RXD — Receive Pin . . . 726
20.3 Memory Map and Register Definition . . . 726
20.3.1 Module Memory Map and Register Definition . . . 726
20.3.2 Register Descriptions . . . 727
20.4 Functional Description . . . 739
20.4.1 Infrared Interface Submodule . . . 740
20.4.2 LIN Support . . . 740
20.4.3 Data Format . . . 741
20.4.4 Baud Rate Generation . . . 742
20.4.5 Transmitter . . . 743
20.4.6 Receiver . . . 748
20.4.7 Single-Wire Operation . . . 756
20.4.8 Loop Operation . . . 757
20.5 Initialization/Application Information . . . 757
20.5.1 Reset Initialization . . . 757
20.5.2 Modes of Operation . . . 757
20.5.3 Interrupt Operation . . . 758
20.5.4 Recovery from Wait Mode . . . 760
20.5.5 Recovery from Stop Mode . . . 760

## Chapter 21
## Serial Peripheral Interface (S12SPIV5)

21.1 Introduction . . . 761
21.1.1 Glossary of Terms . . . 761
21.1.2 Features . . . 761
21.1.3 Modes of Operation . . . 761
21.1.4 Block Diagram . . . 762
21.2 External Signal Description . . . 763
21.2.1 MOSI — Master Out/Slave In Pin . . . 763
21.2.2 MISO — Master In/Slave Out Pin . . . 763
21.2.3 $\overline{SS}$ — Slave Select Pin . . . 764
21.2.4 SCK — Serial Clock Pin . . . 764
21.3 Memory Map and Register Definition . . . 764
21.3.1 Module Memory Map . . . 764
21.3.2 Register Descriptions . . . 765
21.4 Functional Description . . . 773
21.4.1 Master Mode . . . 774
21.4.2 Slave Mode . . . 775
21.4.3 Transmission Formats . . . 776
21.4.4 SPI Baud Rate Generation . . . 781
21.4.5 Special Features . . . 782
21.4.6 Error Conditions . . . 783
21.4.7 Low Power Mode Options . . . 784

# Chapter 22
# Timer Module (TIM16B8CV2) Block Description


22.1 Introduction . . . 787
22.1.1 Features . . . 788
22.1.2 Modes of Operation . . . 788
22.1.3 Block Diagrams . . . 789
22.2 External Signal Description . . . 791
22.2.1 IOC7 — Input Capture and Output Compare Channel 7 Pin . . . 791
22.2.2 IOC6 — Input Capture and Output Compare Channel 6 Pin . . . 791
22.2.3 IOC5 — Input Capture and Output Compare Channel 5 Pin . . . 791
22.2.4 IOC4 — Input Capture and Output Compare Channel 4 Pin . . . 791
22.2.5 IOC3 — Input Capture and Output Compare Channel 3 Pin . . . 791
22.2.6 IOC2 — Input Capture and Output Compare Channel 2 Pin . . . 791
22.2.7 IOC1 — Input Capture and Output Compare Channel 1 Pin . . . 792
22.2.8 IOC0 — Input Capture and Output Compare Channel 0 Pin . . . 792
22.3 Memory Map and Register Definition . . . 792
22.3.1 Module Memory Map . . . 792
22.3.2 Register Descriptions . . . 792
22.4 Functional Description . . . 809
22.4.1 Prescaler . . . 810
22.4.2 Input Capture . . . 811
22.4.3 Output Compare . . . 811
22.4.4 Pulse Accumulator . . . 812
22.4.5 Event Counter Mode . . . 812
22.4.6 Gated Time Accumulation Mode . . . 813
22.5 Resets . . . 813
22.6 Interrupts . . . 813
22.6.1 Channel [7:0] Interrupt (C[7:0]F) . . . 814
22.6.2 Pulse Accumulator Input Interrupt (PAOVI) . . . 814
22.6.3 Pulse Accumulator Overflow Interrupt (PAOVF) . . . 814
22.6.4 Timer Overflow Interrupt (TOF) . . . 814


# Chapter 23
# Voltage Regulator (S12VREGL3V3V1)


23.1 Introduction . . . 815
23.1.1 Features . . . 815
23.1.2 Modes of Operation . . . 815
23.1.3 Block Diagram . . . 816
23.2 External Signal Description . . . 818
23.2.1 VDDR — Regulator Power Input Pins . . . 818
23.2.2 VDDA, VSSA — Regulator Reference Supply Pins . . . 818
23.2.3 VDD, VSS — Regulator Output1 (Core Logic) Pins . . . 818
23.2.4 VDDF — Regulator Output2 (NVM Logic) Pins . . . 819
23.2.5 VDDPLL, VSSPLL — Regulator Output3 (PLL) Pins . . . 819

23.2.6 VDDX — Power Input Pin . . . 819
23.2.7 $V_{REGEN}$ — Optional Regulator Enable Pin . . . 819
23.2.8 $V_{REG_API}$ — Optional Autonomous Periodical Interrupt Output Pin . . . 819
23.3 Memory Map and Register Definition . . . 819
23.3.1 Module Memory Map . . . 820
23.3.2 Register Descriptions . . . 820
23.4 Functional Description . . . 826
23.4.1 General . . . 826
23.4.2 Regulator Core (REG) . . . 826
23.4.3 Low-Voltage Detect (LVD) . . . 826
23.4.4 Power-On Reset (POR) . . . 827
23.4.5 Low-Voltage Reset (LVR) . . . 827
23.4.6 HTD - High Temperature Detect . . . 827
23.4.7 Regulator Control (CTRL) . . . 827
23.4.8 Autonomous Periodical Interrupt (API) . . . 827
23.4.9 Resets . . . 828
23.4.10 Description of Reset Operation . . . 828
23.4.11 Interrupts . . . 828

## Chapter 24
## 128 KByte Flash Module (S12XFTM128K2V1)

24.1 Introduction . . . 832
24.1.1 Glossary . . . 832
24.1.2 Features . . . 833
24.1.3 Block Diagram . . . 834
24.2 External Signal Description . . . 835
24.3 Memory Map and Registers . . . 836
24.3.1 Module Memory Map . . . 836
24.3.2 Register Descriptions . . . 841
24.4 Functional Description . . . 862
24.4.1 Flash Command Operations . . . 862
24.4.2 Flash Command Description . . . 867
24.4.3 Interrupts . . . 887
24.4.4 Wait Mode . . . 888
24.4.5 Stop Mode . . . 888
24.5 Security . . . 888
24.5.1 Unsecuring the MCU using Backdoor Key Access . . . 889
24.5.2 Unsecuring the MCU in Special Single Chip Mode using BDM . . . 890
24.5.3 Mode and Security Effects on Flash Command Availability . . . 890
24.6 Initialization . . . 890

## Chapter 25
## 256 KByte Flash Module (S12XFTM256K2V1)

25.1 Introduction . . . 891

25.1.1 Glossary . . . 892
25.1.2 Features . . . 893
25.1.3 Block Diagram . . . 894
25.2 External Signal Description . . . 895
25.3 Memory Map and Registers . . . 896
25.3.1 Module Memory Map . . . 896
25.3.2 Register Descriptions . . . 901
25.4 Functional Description . . . 922
25.4.1 Flash Command Operations . . . 922
25.4.2 Flash Command Description . . . 927
25.4.3 Interrupts . . . 948
25.4.4 Wait Mode . . . 949
25.4.5 Stop Mode . . . 949
25.5 Security . . . 949
25.5.1 Unsecuring the MCU using Backdoor Key Access . . . 950
25.5.2 Unsecuring the MCU in Special Single Chip Mode using BDM . . . 951
25.5.3 Mode and Security Effects on Flash Command Availability . . . 951
25.6 Initialization . . . 951

## Chapter 26
## 384 KByte Flash Module (S12XFTM384K2V1)

26.1 Introduction . . . 953
26.1.1 Glossary . . . 954
26.1.2 Features . . . 955
26.1.3 Block Diagram . . . 956
26.2 External Signal Description . . . 957
26.3 Memory Map and Registers . . . 958
26.3.1 Module Memory Map . . . 958
26.3.2 Register Descriptions . . . 963
26.4 Functional Description . . . 984
26.4.1 Flash Command Operations . . . 984
26.4.2 Flash Command Description . . . 989
26.4.3 Interrupts . . . 1011
26.4.4 Wait Mode . . . 1012
26.4.5 Stop Mode . . . 1012
26.5 Security . . . 1012
26.5.1 Unsecuring the MCU using Backdoor Key Access . . . 1013
26.5.2 Unsecuring the MCU in Special Single Chip Mode using BDM . . . 1014
26.5.3 Mode and Security Effects on Flash Command Availability . . . 1014
26.6 Initialization . . . 1014

## Chapter 27
## 512 KByte Flash Module (S12XFTM512K3V1)

27.1 Introduction . . . 1016

27.1.1 Glossary . . . 1016
27.1.2 Features . . . 1017
27.1.3 Block Diagram . . . 1018
27.2 External Signal Description . . . 1019
27.3 Memory Map and Registers . . . 1020
27.3.1 Module Memory Map . . . 1020
27.3.2 Register Descriptions . . . 1025
27.4 Functional Description . . . 1046
27.4.1 Flash Command Operations . . . 1046
27.4.2 Flash Command Description . . . 1051
27.4.3 Interrupts . . . 1072
27.4.4 Wait Mode . . . 1073
27.4.5 Stop Mode . . . 1073
27.5 Security . . . 1073
27.5.1 Unsecuring the MCU using Backdoor Key Access . . . 1074
27.5.2 Unsecuring the MCU in Special Single Chip Mode using BDM . . . 1075
27.5.3 Mode and Security Effects on Flash Command Availability . . . 1075
27.6 Initialization . . . 1075

## Chapter 28
## 768 KByte Flash Module (S12XFTM768K4V2)

28.1 Introduction . . . 1077
28.1.1 Glossary . . . 1078
28.1.2 Features . . . 1079
28.1.3 Block Diagram . . . 1080
28.2 External Signal Description . . . 1081
28.3 Memory Map and Registers . . . 1082
28.3.1 Module Memory Map . . . 1082
28.3.2 Register Descriptions . . . 1087
28.4 Functional Description . . . 1108
28.4.1 Flash Command Operations . . . 1108
28.4.2 Flash Command Description . . . 1113
28.4.3 Interrupts . . . 1135
28.4.4 Wait Mode . . . 1136
28.4.5 Stop Mode . . . 1136
28.5 Security . . . 1136
28.5.1 Unsecuring the MCU using Backdoor Key Access . . . 1137
28.5.2 Unsecuring the MCU in Special Single Chip Mode using BDM . . . 1138
28.5.3 Mode and Security Effects on Flash Command Availability . . . 1138
28.6 Initialization . . . 1138

## Chapter 29
## 1024 KByte Flash Module (S12XFTM1024K5V2)

29.1 Introduction . . . 1140

29.1.1 Glossary . . . 1140
29.1.2 Features . . . 1141
29.1.3 Block Diagram . . . 1142
29.2 External Signal Description . . . 1143
29.3 Memory Map and Registers . . . 1144
29.3.1 Module Memory Map . . . 1144
29.3.2 Register Descriptions . . . 1150
29.4 Functional Description . . . 1171
29.4.1 Flash Command Operations . . . 1171
29.4.2 Flash Command Description . . . 1176
29.4.3 Interrupts . . . 1197
29.4.4 Wait Mode . . . 1198
29.4.5 Stop Mode . . . 1198
29.5 Security . . . 1198
29.5.1 Unsecuring the MCU using Backdoor Key Access . . . 1199
29.5.2 Unsecuring the MCU in Special Single Chip Mode using BDM . . . 1200
29.5.3 Mode and Security Effects on Flash Command Availability . . . 1200
29.6 Initialization . . . 1200

## Appendix A
## Electrical Characteristics

A.1 General . . . 1201
A.1.1 Parameter Classification . . . 1201
A.1.2 Power Supply . . . 1201
A.1.3 Pins . . . 1202
A.1.4 Current Injection . . . 1203
A.1.5 Absolute Maximum Ratings . . . 1203
A.1.6 ESD Protection and Latch-up Immunity . . . 1204
A.1.7 Operating Conditions . . . 1206
A.1.8 Power Dissipation and Thermal Characteristics . . . 1207
A.1.9 I/O Characteristics . . . 1209
A.1.10 Supply Currents . . . 1214
A.2 ATD Characteristics . . . 1219
A.2.1 ATD Operating Characteristics . . . 1219
A.2.2 Factors Influencing Accuracy . . . 1219
A.2.3 ATD Accuracy . . . 1221
A.3 NVM, Flash and Emulated EEPROM . . . 1224
A.3.1 Timing Parameters . . . 1224
A.3.2 NVM Reliability Parameters . . . 1231
A.4 Voltage Regulator . . . 1234
A.5 Output Loads . . . 1235
A.5.1 Resistive Loads . . . 1235
A.5.2 Capacitive Loads . . . 1235
A.5.3 Chip Power-up and Voltage Drops . . . 1235
A.6 Reset, Oscillator and PLL . . . 1236

A.6.1 Startup . . . . . 1236
A.6.2 Oscillator . . . . . 1238
A.6.3 Phase Locked Loop . . . . . 1239
A.7 External Interface Timing . . . . . 1241
A.7.1 MSCAN . . . . . 1241
A.7.2 SPI Timing . . . . . 1241
A.7.3 External Bus Timing . . . . . 1247

## Appendix B
## Package Information

B.1 208 MAPBGA . . . . . 1259
B.2 144-Pin LQFP . . . . . 1259
B.3 112-Pin LQFP Package . . . . . 1261
B.4 80-Pin QFP Package . . . . . 1262

## Appendix C
## PCB Layout Guidelines

## Appendix D
## Derivative Differences

D.1 Memory Sizes and Package Options S12XE - Family . . . . . 1268
D.2 Pinout explanations: . . . . . 1270

## Appendix E
## Detailed Register Address Map

## Appendix F
## Ordering Information

# Chapter 1
# Device Overview MC9S12XE-Family

## 1.1 Introduction

The MC9S12XE-Family of micro controllers is a further development of the S12XD-Family including new features for enhanced system integrity and greater functionality. These new features include a Memory Protection Unit (MPU) and Error Correction Code (ECC) on the Flash memory together with enhanced EEPROM functionality (EEE), an enhanced XGATE, an Internally filtered, frequency modulated Phase Locked Loop (IPLL) and an enhanced ATD. The E-Family extends the S12X product range up to 1MB of Flash memory with increased I/O capability in the 208-pin version of the flagship MC9S12XE100.

The MC9S12XE-Family delivers 32-bit performance with all the advantages and efficiencies of a 16 bit MCU. It retains the low cost, power consumption, EMC and code-size efficiency advantages currently enjoyed by users of Freescale's existing 16-Bit MC9S12 and S12X MCU families. There is a high level of compatibility between the S12XE and S12XD families.

The MC9S12XE-Family features an enhanced version of the performance-boosting XGATE co-processor which is programmable in "C" language and runs at twice the bus frequency of the S12X with an instruction set optimized for data movement, logic and bit manipulation instructions and which can service any peripheral module on the device. The new enhanced version has improved interrupt handling capability and is fully compatible with the existing XGATE module.

The MC9S12XE-Family is composed of standard on-chip peripherals including up to 64Kbytes of RAM, eight asynchronous serial communications interfaces (SCI), three serial peripheral interfaces (SPI), an 8-channel IC/OC enhanced capture timer (ECT), two 16-channel, 12-bit analog-to-digital converters, an 8-channel pulse-width modulator (PWM), five CAN 2.0 A, B software compatible modules (MSCAN12), two inter-IC bus blocks (IIC), an 8-channel 24-bit periodic interrupt timer (PIT) and an 8-channel 16-bit standard timer module (TIM).

The MC9S12XE-Family uses 16-bit wide accesses without wait states for all peripherals and memories. The non-multiplexed expanded bus interface available on the 144/208-Pin versions allows an easy interface to external memories.

In addition to the I/O ports available in each module, up to 26 further I/O ports are available with interrupt capability allowing Wake-Up from STOP or WAIT modes. The MC9S12XE-Family is available in 208-Pin MAPBGA, 144-Pin LQFP, 112-Pin LQFP or 80-Pin QFP options.

### 1.1.1 Features

Features of the MC9S12XE-Family are listed here. Please see Table D-2.for memory options and Table D-2. for the peripheral features that are available on the different family members.

- 16-Bit CPU12X
  - Upward compatible with MC9S12 instruction set with the exception of five Fuzzy instructions (MEM, WAV, WAVR, REV, REVW) which have been removed
  - Enhanced indexed addressing
  - Access to large data segments independent of PPAGE
- INT (interrupt module)
  - Eight levels of nested interrupts
  - Flexible assignment of interrupt sources to each interrupt level.
  - External non-maskable high priority interrupt (XIRQ)
  - Internal non-maskable high priority Memory Protection Unit interrupt
  - Up to 24 pins on ports J, H and P configurable as rising or falling edge sensitive interrupts
- EBI (external bus interface)(available in 208-Pin and 144-Pin packages only)
  - Up to four chip select outputs to select 16K, 1M, 2M and up to 4MByte address spaces
  - Each chip select output can be configured to complete transaction on either the time-out of one of the two wait state generators or the deassertion of EWAIT signal
- MMC (module mapping control)
- DBG (debug module)
  - Monitoring of CPU and/or XGATE busses with tag-type or force-type breakpoint requests
  - 64 x 64-bit circular trace buffer captures change-of-flow or memory access information
- BDM (background debug mode)
- MPU (memory protection unit)
  - 8 address regions definable per active program task
  - Address range granularity as low as 8-bytes
  - No write / No execute Protection Attributes
  - Non-maskable interrupt on access violation
- XGATE
  - Programmable, high performance I/O coprocessor module
  - Transfers data to or from all peripherals and RAM without CPU intervention or CPU wait states
  - Performs logical, shifts, arithmetic, and bit operations on data
  - Can interrupt the HCS12X CPU signalling transfer completion
  - Triggers from any hardware module as well as from the CPU possible
  - Two interrupt levels to service high priority tasks
  - Hardware support for stack pointer initialisation
- OSC_LCP (oscillator)
  - Low power loop control Pierce oscillator utilizing a 4MHz to 16MHz crystal
  - Good noise immunity
  - Full-swing Pierce option utilizing a 2MHz to 40MHz crystal
  - Transconductance sized for optimum start-up margin for typical crystals
- IPLL (Internally filtered, frequency modulated phase-locked-loop clock generation)

  - No external components required
  - Configurable option to spread spectrum for reduced EMC radiation (frequency modulation)
- CRG (clock and reset generation)
  - COP watchdog
  - Real time interrupt
  - Clock monitor
  - Fast wake up from STOP in self clock mode
- Memory Options
  - 128K, 256k, 384K, 512K, 768K and 1M byte Flash
  - 2K, 4K byte emulated EEPROM
  - 12K, 16K, 24K, 32K, 48K and 64K Byte RAM
- Flash General Features
  - 64 data bits plus 8 syndrome ECC (Error Correction Code) bits allow single bit failure correction and double fault detection
  - Erase sector size 1024 bytes
  - Automated program and erase algorithm
- D-Flash Features
  - Up to 32 Kbytes of D-Flash memory with 256 byte sectors for user access.
  - Dedicated commands to control access to the D-Flash memory over EEE operation.
  - Single bit fault correction and double bit fault detection within a word during read operations.
  - Automated program and erase algorithm with verify and generation of ECC parity bits.
  - Fast sector erase and word program operation.
  - Ability to program up to four words in a burst sequence
- Emulated EEPROM Features
  - Automatic EEE file handling using an internal Memory Controller.
  - Automatic transfer of valid EEE data from D-Flash memory to buffer RAM on reset.
  - Ability to monitor the number of outstanding EEE related buffer RAM words left to be programmed into D-Flash memory.
  - Ability to disable EEE operation and allow priority access to the D-Flash memory.
  - Ability to cancel all pending EEE operations and allow priority access to the D-Flash memory.
- Two 16-channel, 12-bit Analog-to-Digital Converters
  - 8/10/12 Bit resolution
  - 3µs, 10-bit single conversion time
  - Left/right, signed/unsigned result data
  - External and internal conversion trigger capability
  - Internal oscillator for conversion in Stop modes
  - Wake from low power modes on analog comparison > or <= match
- Five MSCAN (1 M bit per second, CAN 2.0 A, B software compatible modules)
  - Five receive and three transmit buffers

  - Flexible identifier filter programmable as 2 x 32 bit, 4 x 16 bit, or 8 x 8 bit
  - Four separate interrupt channels for Rx, Tx, error, and wake-up
  - Low-pass filter wake-up function
  - Loop-back for self-test operation
- ECT (enhanced capture timer)
  - 8 x 16-bit channels for input capture or output compare
  - 16-bit free-running counter with 8-bit precision prescaler
  - 16-bit modulus down counter with 8-bit precision prescaler
  - Four 8-bit or two 16-bit pulse accumulators
- TIM (standard timer module)
  - 8 x 16-bit channels for input capture or output compare
  - 16-bit free-running counter with 8-bit precision prescaler
  - 1 x 16-bit pulse accumulator
- PIT (periodic interrupt timer)
  - Up to eight timers with independent time-out periods
  - Time-out periods selectable between 1 and $2^{24}$ bus clock cycles
  - Time-out interrupt and peripheral triggers
- 8 PWM (pulse-width modulator) channels
  - 8 channel x 8-bit or 4 channel x 16-bit Pulse Width Modulator
  - programmable period and duty cycle per channel
  - Center- or left-aligned outputs
  - Programmable clock select logic with a wide range of frequencies
  - Fast emergency shutdown input
- Three Serial Peripheral Interface Modules (SPI)
  - Configurable for 8 or 16-bit data size
- Eight Serial Communication Interfaces (SCI)
  - Standard mark/space non-return-to-zero (NRZ) format
  - Selectable IrDA 1.4 return-to-zero-inverted (RZI) format with programmable pulse widths
- Two Inter-IC bus (IIC) Modules
  - Multi-master operation
  - Software programmable for one of 256 different serial clock frequencies
  - Broadcast mode support
  - 10-bit address support
- On-Chip Voltage Regulator
  - Two parallel, linear voltage regulators with bandgap reference
  - Low-voltage detect (LVD) with low-voltage interrupt (LVI)
  - Power-on reset (POR) circuit
  - 3.3V and 5V range operation
  - Low-voltage reset (LVR)

- Low-power wake-up timer (API)
  - Available in all modes including Full Stop Mode
  - Trimmable to +-5% accuracy
  - Time-out periods range from 0.2ms to ~13s with a 0.2ms resolution
- Input/Output
  - Up to 152 general-purpose input/output (I/O) pins plus 2 input-only pins
  - Hysteresis and configurable pull up/pull down device on all input pins
  - Configurable drive strength on all output pins
- Package Options
  - 208-pin MAPBGA
  - 144-pin low-profile quad flat-pack (LQFP)
  - 112-pin low-profile quad flat-pack (LQFP)
  - 80-pin quad flat-pack (QFP)
- 50MHz maximum CPU bus frequency, 100MHz maximum XGATE bus frequency

### 1.1.2 Modes of Operation

Memory map and bus interface modes:

- Normal and emulation operating modes
  - Normal single-chip mode
  - Normal expanded mode
  - Emulation of single-chip mode
  - Emulation of expanded mode
- Special Operating Modes
  - Special single-chip mode with active background debug mode
  - Special test mode (**Freescale use only**)

Low-power modes:

- System stop modes
  - Pseudo stop mode
  - Full stop mode with fast wake-up option
- System wait mode

Operating system states

- Supervisor state
- User state

## 1.1.3 Block Diagram

Figure 1-1 shows a block diagram of the MC9S12XE-Family devices

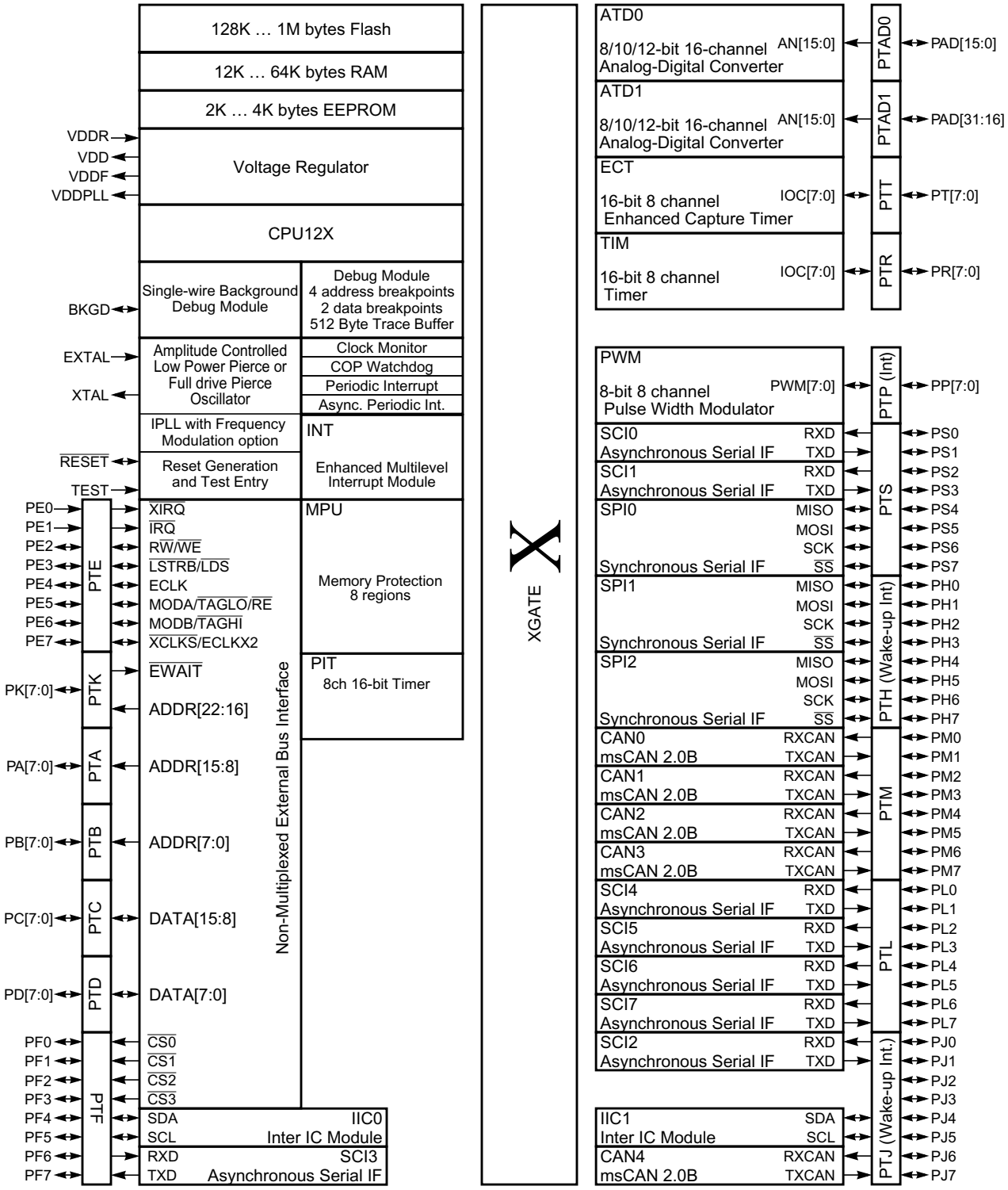



**Figure 1-1. MC9S12XE-Family Block Diagram**

### 1.1.4 Device Memory Map

Table 1-1 shows the device register memory map.

**Table 1-1. Device Register Memory Map**

| Address | Module | Size (Bytes) |
|---|---|---|
| 0x0000–0x0009 | PIM (port integration module) | 10 |
| 0x000A–0x000B | MMC (memory map control) | 2 |
| 0x000C–0x000D | PIM (port integration module) | 2 |
| 0x000E–0x000F | EBI (external bus interface) | 2 |
| 0x0010–0x0017 | MMC (memory map control) | 8 |
| 0x0018–0x0019 | Reserved | 2 |
| 0x001A–0x001B | Device ID register | 2 |
| 0x001C–0x001F | PIM (port integration module) | 4 |
| 0x0020–0x002F | DBG (debug module) | 16 |
| 0x0030–0x0031 | Reserved | 2 |
| 0x0032–0x0033 | PIM (port integration module) | 2 |
| 0x0034–0x003F | ECRG (clock and reset generator) | 12 |
| 0x0040–0x007F | ECT (enhanced capture timer 16-bit 8-channel)s | 64 |
| 0x0080–0x00AF | ATD1 (analog-to-digital converter 12-bit 16-channel) | 48 |
| 0x00B0–0x00B7 | IIC1 (inter IC bus) | 8 |
| 0x00B8–0x00BF | SCI2 (serial communications interface) | 8 |
| 0x00C0–0x00C7 | SCI3 (serial communications interface) | 8 |
| 0x00C8–0x00CF | SCI0 (serial communications interface) | 8 |
| 0x00D0–0x00D7 | SCI1 (serial communications interface) | 8 |
| 0x00D8–0x00DF | SPI0 (serial peripheral interface) | 8 |
| 0x00E0–0x00E7 | IIC0 (inter IC bus) | 8 |
| 0x00E8–0x00EF | Reserved | 8 |
| 0x00F0–0x00F7 | SPI1 (serial peripheral interface) | 8 |
| 0x00F8–0x00FF | SPI2 (serial peripheral interface) | 8 |
| 0x0100–0x0113 | FTM control registers | 20 |
| 0x0114–0x011F | MPU (memory protection unit) | 12 |
| 0x0120–0x012F | INT (interrupt module) | 16 |
| 0x0130–0x0137 | SCI4 (serial communications interface) | 8 |
| 0x0138–0x013F | SCI5 (serial communications interface) | 8 |
| 0x0140–0x017F | CAN0 | 64 |
| 0x0180–0x01BF | CAN1 | 64 |
| 0x01C0–0x01FF | CAN2 | 64 |

**Table 1-1. Device Register Memory Map (continued)**

| Address | Module | Size (Bytes) |
|---|---|---|
| 0x0200–0x023F | CAN3 | 64 |
| 0x0240–0x027F | PIM (port integration module) | 64 |
| 0x0280–0x02BF | CAN4 | 64 |
| 0x02C0–0x02EF | ATD0 (analog-to-digital converter 12 bit 16-channel) | 48 |
| 0x02F0–0x02F7 | Voltage regulator | 8 |
| 0x02F8–0x02FF | Reserved | 8 |
| 0x0300–0x0327 | PWM (pulse-width modulator 8 channels) | 40 |
| 0x0328–0x032F | Reserved | 8 |
| 0x0330–0x0337 | SCI6 (serial communications interface) | 8 |
| 0x0338–0x033F | SCI7 (serial communications interface) | 8 |
| 0x0340–0x0367 | PIT (periodic interrupt timer) | 40 |
| 0x0368–0x037F | PIM (port integration module) | 24 |
| 0x0380–0x03BF | XGATE | 64 |
| 0x03C0–0x03CF | Reserved | 16 |
| 0x03D0–0x03FF | TIM (timer module) | 48 |
| 0x0400–0x07FF | Reserved | 1024 |

**NOTE**

Reserved register space shown in Table 1-1 is not allocated to any module. This register space is reserved for future use. Writing to these locations have no effect. Read access to these locations returns zero.

### 1.1.5 Address Mapping

Figure 1-2 shows S12XE CPU & BDM local address translation to the global memory map. It indicates also the location of the internal resources in the memory map.

EEEPROM size is presented like a fixed 256 KByte in the memory map.

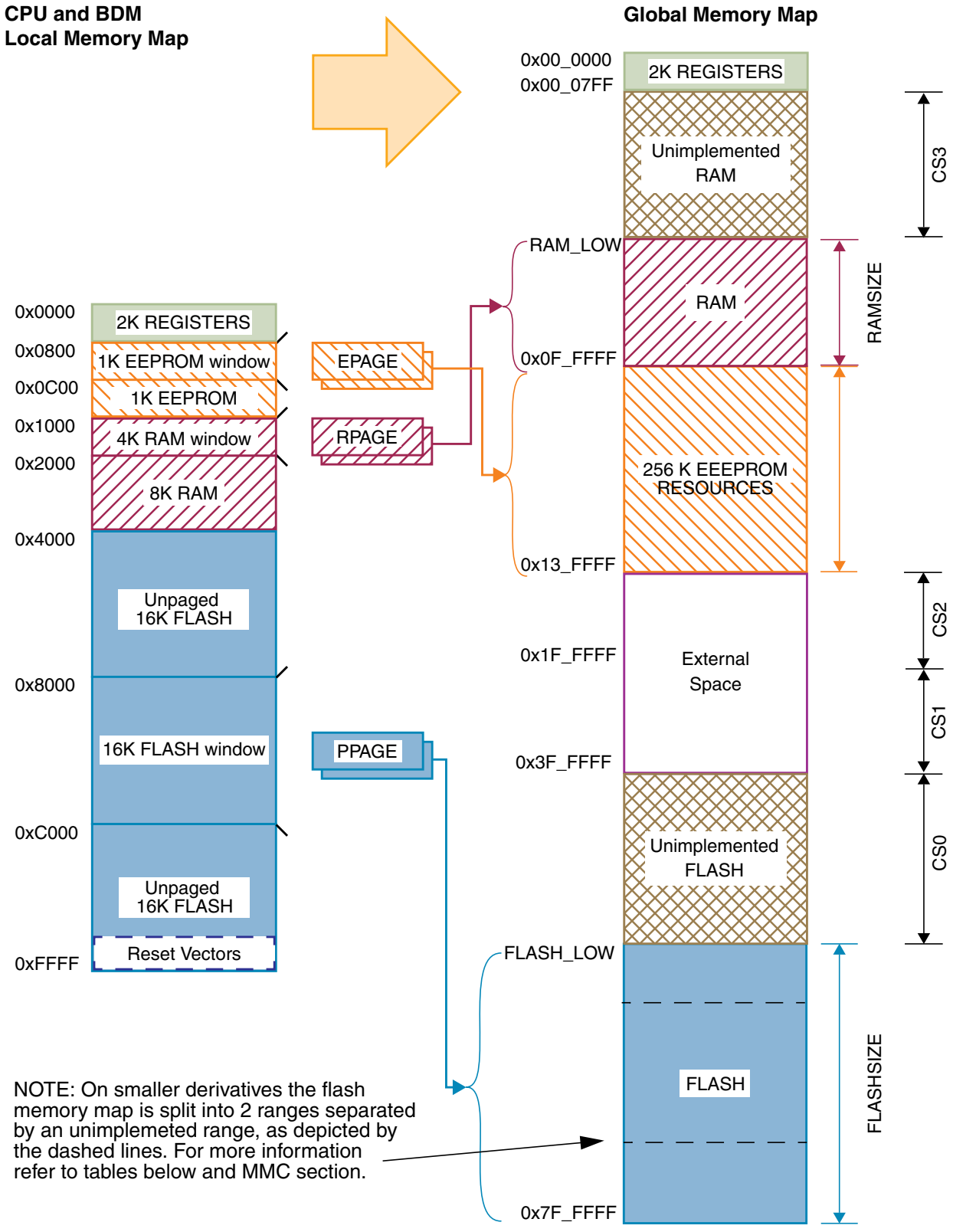



**Figure 1-2. MC9S12XE100 Global Memory Map**

Unimplemented RAM pages are mapped externally in expanded modes. Accessing unimplemented RAM pages in single chip modes causes an illegal address reset if the MPU is not configured to flag an MPU protection error in that range.

Accessing unimplemented FLASH pages in single chip modes causes an illegal address reset if the MPU is not configured to flag an MPU protection error in that range.

The PARTID value should be referenced regarding the specific memory map for any given device. For devices sharing the same part ID, the memory regions which are implemented on the larger device but not supported on the smaller device are implemented but untested on that smaller device. These regions do not appear as unimplemented in the memory map and do not result in an illegal address reset if erroneously accessed.

**Table 1-2. Unimplemented Range Mapping to Part ID**

| Part ID | RAM_LOW | EE_LOW | Flash Blocks | Registers |
|---|---|---|---|---|
| 0xCC8x | 0x0F_0000 | 0x13_F000 | B3, B2, B1S, B1N, B0 | 2K |
| 0xCC9x | 0x0F_0000 | 0x13_F000 | B3, B2, B1S, B1N, B0 | 2K |
| 0xC48x | 0x0F_8000 | 0x13_F000 | B1N, B1S, B0 | 2K |
| 0xC08x | 0x0F_C000 | 0x13_F000 | B1S, B0(128K) | 2K |

From the above the following examples can be derived.

The 9S12XEP768 is currently only available as a 9S12XEP100 die, thus the unimplemented FLASH pages are those of the 9S12XEP100 device map.

The 9S12XEQ384, 9S12XEG384, 9S12XES384 are currently only available as a 9S12XEQ512 die, thus the unimplemented FLASH pages are those of the 9S12XEQ512 device map.

The 9S12XEG128 is currently only available as a 9S12XET256 die, thus the unimplemented FLASH pages are those of the 9S12XET256 device map.

The range between 0x10_0000 and 0x13_FFFF is mapped to EEPROM resources. The actual EEPROM and dataflash block sizes are listed in Table 1-4. Within EEPROM resource range an address range exists which is neither used by EEPROM resources nor remapped to external resources via chip selects (see the FTM/MMC descriptions for details). These ranges do not constitute unimplemented areas.

Accessing reserved registers within the 2K register space does not generate an illegal address reset.

The fixed 8K RAM default location in the global map is 0x0F_E000- 0x0F_FFFF. This is subject to remapping when configuring the local address map for a larger RAM access range.

Figure 1-3 shows XGATE local address translation to the global memory map. It indicates also the location of used internal resources in the memory map.

**Table 1-3. XGATE Resources**

| Internal Resource | Size /KByte | $Address |
|---|---|---|
| XGATE RAM | 32K | XGRAM_LOW = 0x0F_8000 |
| FLASH | 30K(1) | XGFLASH_HIGH = 0x78_8000 |

1. This value is calculated by the following formula: (64K -2K- XGRAMSIZE)

**Table 1-4. Derivative Dependent Memory Parameters**

| Device | FLASH_LOW | PPAGE (1) | RAM_LOW | RPAGE (2) | EE_LOW | EPAGE |
|---|---|---|---|---|---|---|
| 9S12XEP100 | 0x70_0000 | 64 | 0x0F_0000 | 16 | 0x13_F000 | 4(3) + 32(4) |
| 9S12XEP768 | 0x74_0000 | 48 | 0x0F_4000 | 12 | 0x13_F000 | 4 + 32 |
| 9S12XEQ512 | 0x78_0000 | 32 | 0x0F_8000 | 8 | 0x13_F000 | 4 + 32 |
| 9S12XEx384 | 0x78_0000(5) | 24 | 0x0F_A000 | 6 | 0x13_F000 | 4 + 32 |
| 9S12XET256<br>9S12XEA256 (6) | 0x78_0000(7) | 16 | 0x0F_C000 | 4 | 0x13_F000 | 4 + 32 |
| 9S12XEG128<br>9S12XEA128[6] | 0x78_0000(8) | 8 | 0x0F_D000 | 3 | 0x13_F800 | 2 + 32 |

1. Number of 16K pages addressable via PPAGE register
2. Number of 4K pages addressing the RAM. RAM can also be mapped to 0x4000 - 0x7FFF
3. Number of 1K pages addressing the Cache RAM via the EPAGE register counting downwards from 0xFF
4. Number of 1K pages addressing the Data flash via the EPAGE register starting upwards from 0x00
5. The 384K memory map is split into a 128K block from 0x78_0000 to 0x79_FFFF and a 256K block from 0x7C_0000 to 0x7F_FFFF
6. The 9S12XEA devices are a special bondout for access to extra ADC channels in 80QFP. Available in 80QFP only. WARNING: NOT PIN-COMPATIBLE WITH REST OF FAMILY.
7. The 256K memory map is split into a 128K block from 0x78_0000 to 0x79_FFFF and a 128K block from 0x7E_0000 to 0x7F_FFFF
8. The 128K memory map is split into a 64K block from 0x78_0000 to 0x78_FFFF and a 64K block from 0x7F_0000 to 0x7F_FFFF

**Table 1-5. Derivative Dependent Flash Block Mapping**

| Device | 0x70_0000 | 0x74_0000 | 0x78_0000 | 0x7A_0000 | 0x7C_0000 | 0x7E_0000 |
|---|---|---|---|---|---|---|
| 9S12XEP100 | B3 | B2 | B1S | B1N | B0 | |
| 9S12XEP768 | — | B2 | B1S | B1N | B0 | |
| 9S12XEQ512 | — | — | B1S | B1N | B0 | |
| 9S12XEx384 | — | — | B1S | — | B0 | |

**Table 1-5. Derivative Dependent Flash Block Mapping (continued)**

| Device | 0x70_0000 | 0x74_0000 | 0x78_0000 | 0x7A_0000 | 0x7C_0000 | 0x7E_0000 |
|---|---|---|---|---|---|---|
| 9S12XET256<br>9S12XEA256 (1) | — | — | B1S | — | — | B0(128K) |
| 9S12XEG128<br>9S12XEA128[1] | — | — | B1S (64K) | — | — | B0 (64K) |

1. The 9S12XEA devices are special bondouts for access to extra ADC channels in 80QFP. Available in 80QFP only. WARNING: NOT PIN-COMPATIBLE WITH REST OF FAMILY.

Block B1 is divided into two 128K blocks. The XGATE is always mapped to block B1S.

On the 9S12XEG128 the flash is divided into two 64K blocks B0 and B1S, the B1S range extending from 0x78_0000 to 0x78_FFFF, the B0 range extending from 0x7F_0000 to 0x7F_FFFF.

The block B0 is a reduced size 128K block on the 256K derivative. On the larger derivatives B0 is a 256K block. The block B0 is a reduced size 64K block on the 128K derivative.

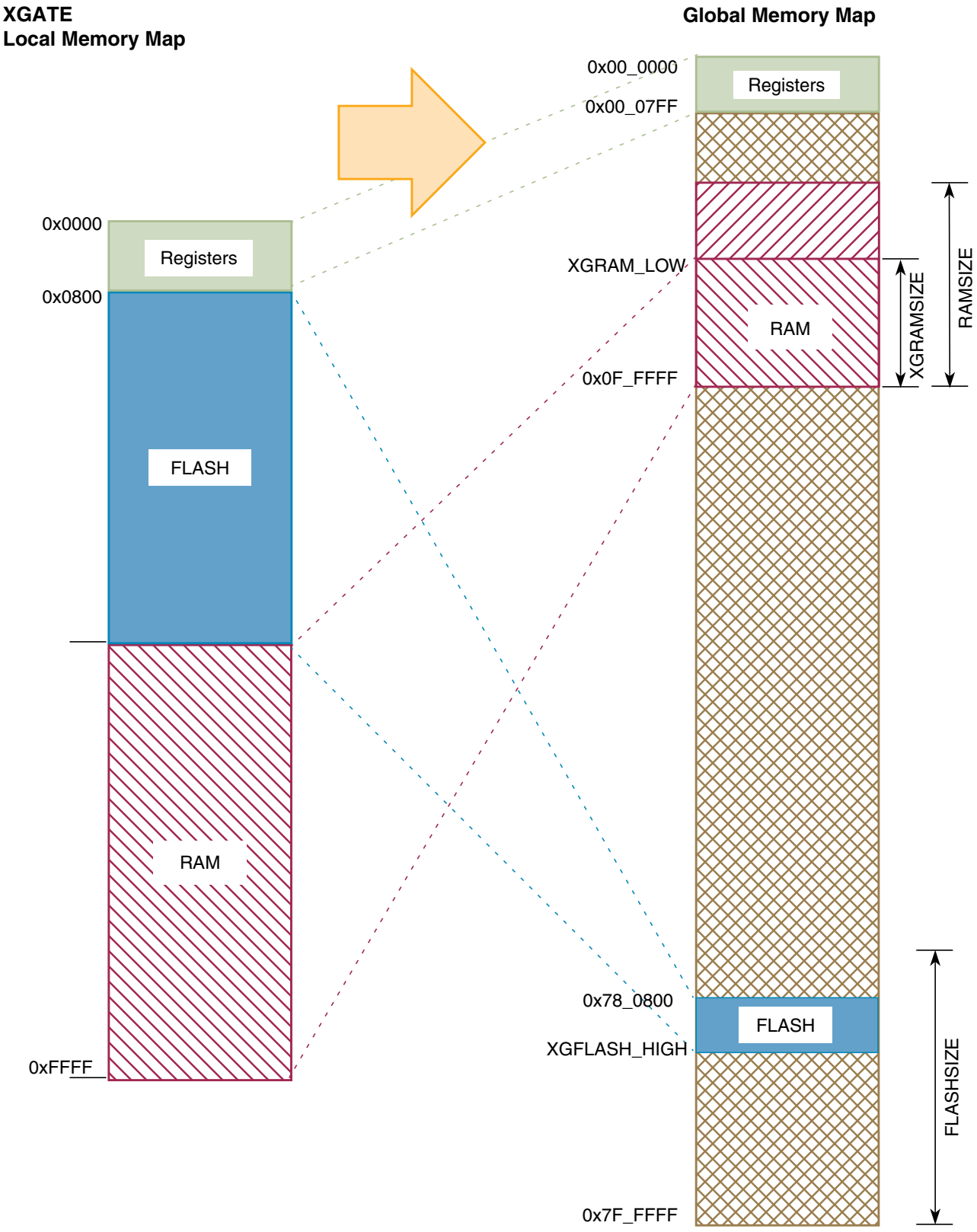



**Figure 1-3. XGATE Global Address Mapping**

### 1.1.6 Detailed Register Map

The detailed register map is listed in Appendix A.

### 1.1.7 Part ID Assignments

The part ID is located in two 8-bit registers PARTIDH and PARTIDL (addresses 0x001A and 0x001B). The read-only value is a unique part ID for each revision of the chip. Table 1-6 shows the assigned part ID number and Mask Set number.

The Version ID is a word located in a flash information row at 0x40_00E8. The version ID number indicates a specific version of internal NVM variables used to patch NVM errata.
The default is no patch (0xFFFF).

**Table 1-6. Assigned Part ID Numbers**

| Device | Mask Set Number | Part ID[1] | Version ID |
|---|---|---|---|
| MC9S12XEP100 | 0M22E | 0xCC80 | 0xFFFF |
| MC9S12XEP100 | 1M22E | 0xCC80 | 0xFFFF |
| MC9S12XEP100 | 2M22E | 0xCC82 | 0xFFFF |
| MC9S12XEP100 | 0M48H | 0xCC90 | 0xFFFF |
| MC9S12XEP100 | 1M48H | 0xCC91 | 0xFFFF |
| MC9S12XEP100 | 2M48H | 0xCC92 | 0xFFFF |
| MC9S12XEP100 | 3M48H | 0xCC93 | 0xFFFF |
| MC9S12XEP100, MC9S12XEP768[2] | 4M48H | 0xCC94 | 0xFFFF |
| MC9S12XEP100, MC9S12XEP768[2] | 5M48H | 0xCC94 | 0x0004 |
| MC9S12XEP100, MC9S12XEP768[2] | 0N35H | 0xCC95 | 0xFFFF |
| MC9S12XEP100, MC9S12XEP768[2] | 1N35H | 0xCC95 | 0x0004 |
| MC9S12XEQ512 | 0M25J | 0xC480 | 0xFFFF |
| MC9S12XEQ512 | 1M25J | 0xC481 | 0xFFFF |
| MC9S12XEQ512, MC9S12XET512 | 2M25J | 0xC482 | 0xFFFF |
| MC9S12XEQ512, MC9S12XET512 | 3M25J | 0xC482 | 0x0004 |
| MC9S12XEQ384[3], MC9S12XEG384[3], MC9S12XES384[3] | 2M25J | 0xC482 | 0xFFFF |
| MC9S12XEQ384[3], MC9S12XEG384[3], MC9S12XES384[3] | 3M25J | 0xC482 | 0x0004 |
| MC9S12XEQ512, MC9S12XET512 | 0M12S | 0xC483 | 0xFFFF |
| MC9S12XEQ512, MC9S12XET512 | 1M12S | 0xC483 | 0x0004 |
| MC9S12XEQ384[3], MC9S12XEG384[3], MC9S12XES384[3] | 0M12S | 0xC483 | 0xFFFF |
| MC9S12XEQ384[3], MC9S12XEG384[3], MC9S12XES384[3] | 1M12S | 0xC483 | 0x0004 |
| MC9S12XET256, MC9S12XEG256 | 0M53J | 0xC080 | 0xFFFF |
| MC9S12XET256, MC9S12XEG256, MC9S12XEA256 | 1M53J | 0xC081 | 0xFFFF |
| MC9S12XET256, MC9S12XEG256, MC9S12XEA256 | 2M53J | 0xC081 | 0x0004 |
| MC9S12XEG128[4], MC9S12XEA128[4] | 1M53J | 0xC081 | 0xFFFF |
| MC9S12XEG128[4], MC9S12XEA128[4] | 2M53J | 0xC081 | 0x0004 |
| MC9S12XET256, MC9S12XEG256, MC9S12XEA256 | 0N36H | 0xC082 | 0xFFFF |
| MC9S12XET256, MC9S12XEG256, MC9S12XEA256 | 1N36H | 0xC082 | 0x0004 |
| MC9S12XEG128[4], MC9S12XEA128[4] | 0N36H | 0xC082 | 0xFFFF |
| MC9S12XEG128[4], MC9S12XEA128[4] | 1N36H | 0xC082 | 0x0004 |

1. The coding is as follows:
   Bit 15-12: Major family identifier
   Bit 11-6: Minor family identifier
   Bit 5-4: Major mask set revision number including FAB transfers
   Bit 3-0: Minor — non full — mask set revision
2. Currently available as MC9S12XEP100 die only
3. Currently available as MC9S12XEQ512 die only
4. Currently available as MC9S12XET256 die only

## 1.2 Signal Description

This section describes signals that connect off-chip. It includes a pinout diagram, a table of signal properties, and detailed discussion of signals. It is built from the signal description sections of the Block User Guides of the individual IP blocks on the device.

### 1.2.1 Device Pinout

The MC9S12XE-Family offers pin-compatible packaged devices to assist with system development and accommodate expansion of the application.

**NOTE**

Smaller derivatives within the MC9S12XE-Family feature a subset of the listed modules. Refer to Appendix D Derivative Differences for more information about derivative device module subset and to Table 1-7. Port Availability by Package Option and Table 1-9. Pin-Out Summary for details of pins available in different package options.

The MC9S12XE-Family devices are offered in the following package options:

- 208-pin MAPBGA package with an external bus interface (address/data bus)
- 144-pin LQFP package with an external bus interface (address/data bus)
- 112-pin LQFP without external bus interface
- 80-pin QFP without external bus interface

| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| A | N.C. | N.C. | PP7 | PM0 | PM1 | PF5 | PF3 | PF1 | PJ6 | PS6 | PS5 | PS3 | PM6 | PAD19 | N.C. | N.C. |
| B | N.C. | PP2 | PP6 | PF7 | PF6 | PF4 | PF2 | PF0 | TEST | PS4 | PS1 | PAD23 | PAD21 | PAD18 | PAD31 | N.C. |
| C | PJ2 | PP1 | PP4 | PP5 | PK7 | PM2 | PM4 | PJ5 | PS7 | PS2 | PM7 | PAD20 | VRL | PAD16 | PAD07 | PAD14 |
| D | PK1 | PJ3 | PP0 | PP3 | VDDX | PM3 | PM5 | PJ4 | PJ7 | VDDX | PS0 | PAD22 | VRH | PAD17 | PAD30 | PAD29 |
| E | PK0 | PK3 | PK2 | PK6 | | | | | | | | | VSSA | PAD15 | PAD06 | PAD28 |
| F | PR1 | PR0 | PT0 | VDDX | | | | | | | | | VDDA | PAD05 | PAD13 | PAD27 |
| G | PT2 | PT3 | PR2 | PT1 | | | VSSX | VSSX | VSSX | VSSX | | | VDDA | PAD12 | PAD04 | PAD11 |
| H | PR3 | PR4 | PT4 | VDDF | | | VSSX | VSSX | VSSX | VSSX | | | VSSA | PAD26 | PAD03 | PAD10 |
| J | PT5 | PR5 | PT6 | VSS1 | | | VSSX | VSSX | VSSX | VSSX | | | VSS2 | PAD09 | PAD25 | PAD02 |
| K | PR6 | PT7 | PK4 | PR7 | | | VSSX | VSSX | VSSX | VSSX | | | VDD | PD7 | PAD24 | PAD01 |
| L | PK5 | PJ1 | BKGD | VDDX | | | | | | | | | VDDX | PD4 | PAD00 | PAD08 |
| M | PJ0 | PC0 | PB1 | PC1 | | | | | | | | | PA6 | PA2 | PD5 | PD6 |
| N | PC2 | PC3 | PB2 | PC7 | PL1 | PE6 | VDDX | VDDR | VSS3 | PH3 | PH1 | VDDX | PE1 | PA1 | PA5 | PA7 |
| P | PB0 | PB3 | PB4 | PC4 | PL2 | PL0 | PE4 | RESET | PL7 | PL6 | PH0 | PE2 | PE0 | PA0 | PA3 | PA4 |
| R | N.C. | PB5 | PB6 | PB7 | PC6 | PH6 | PH4 | PE5 | VSS PLL | VDD PLL | PH2 | PL4 | PD1 | PD3 | PE3 | N.C. |
| T | N.C. | N.C. | PC5 | PL3 | PH7 | PH5 | PE7 | VSS PLL | EXTAL | XTAL | VDD PLL | PL5 | PD0 | PD2 | N.C. | N.C. |

**Figure 1-4. - Pin Assignments, 208 MAPBGA Package**

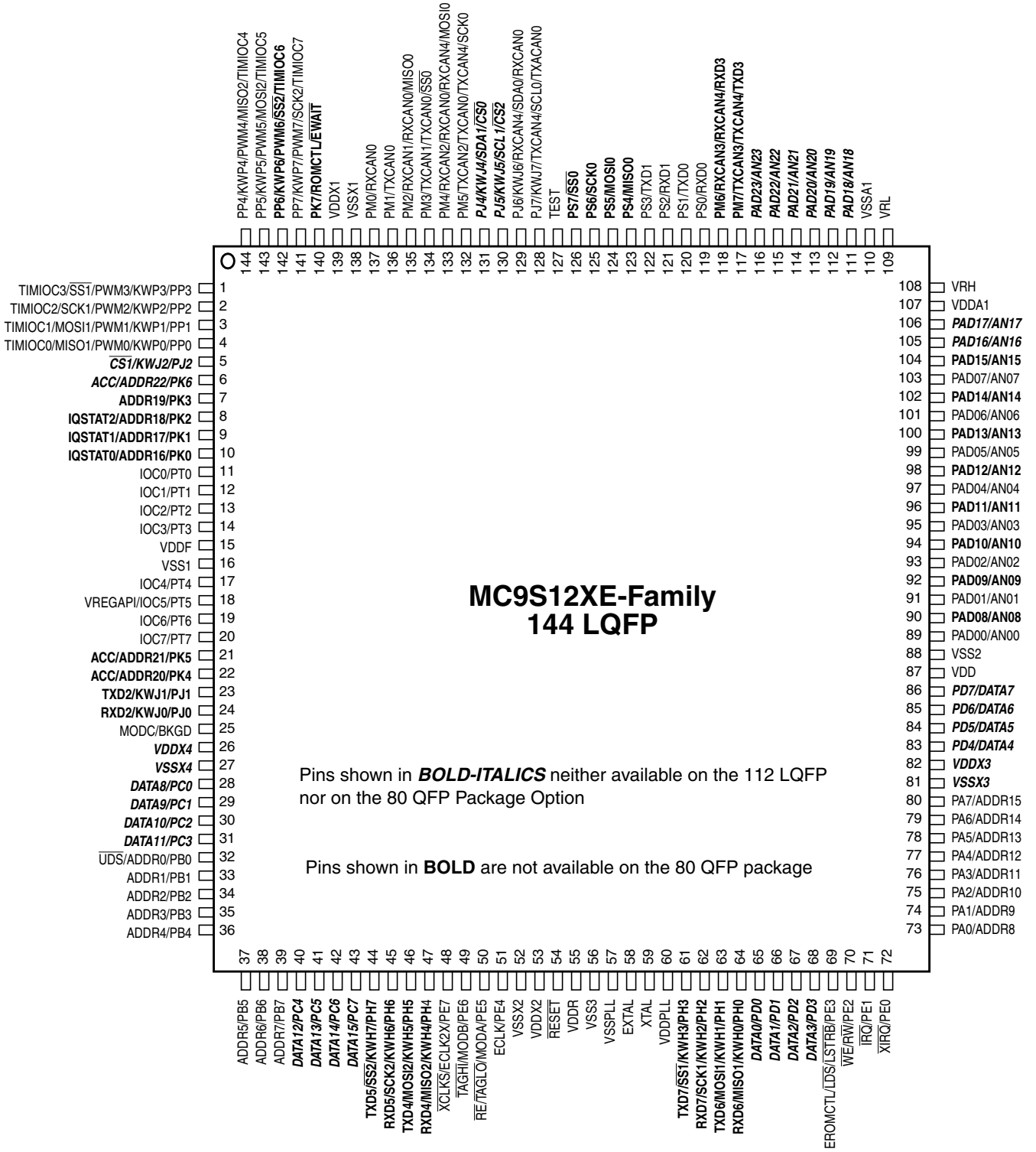



**Figure 1-5. MC9S12XE-Family Pin Assignments 144-pin LQFP Package**

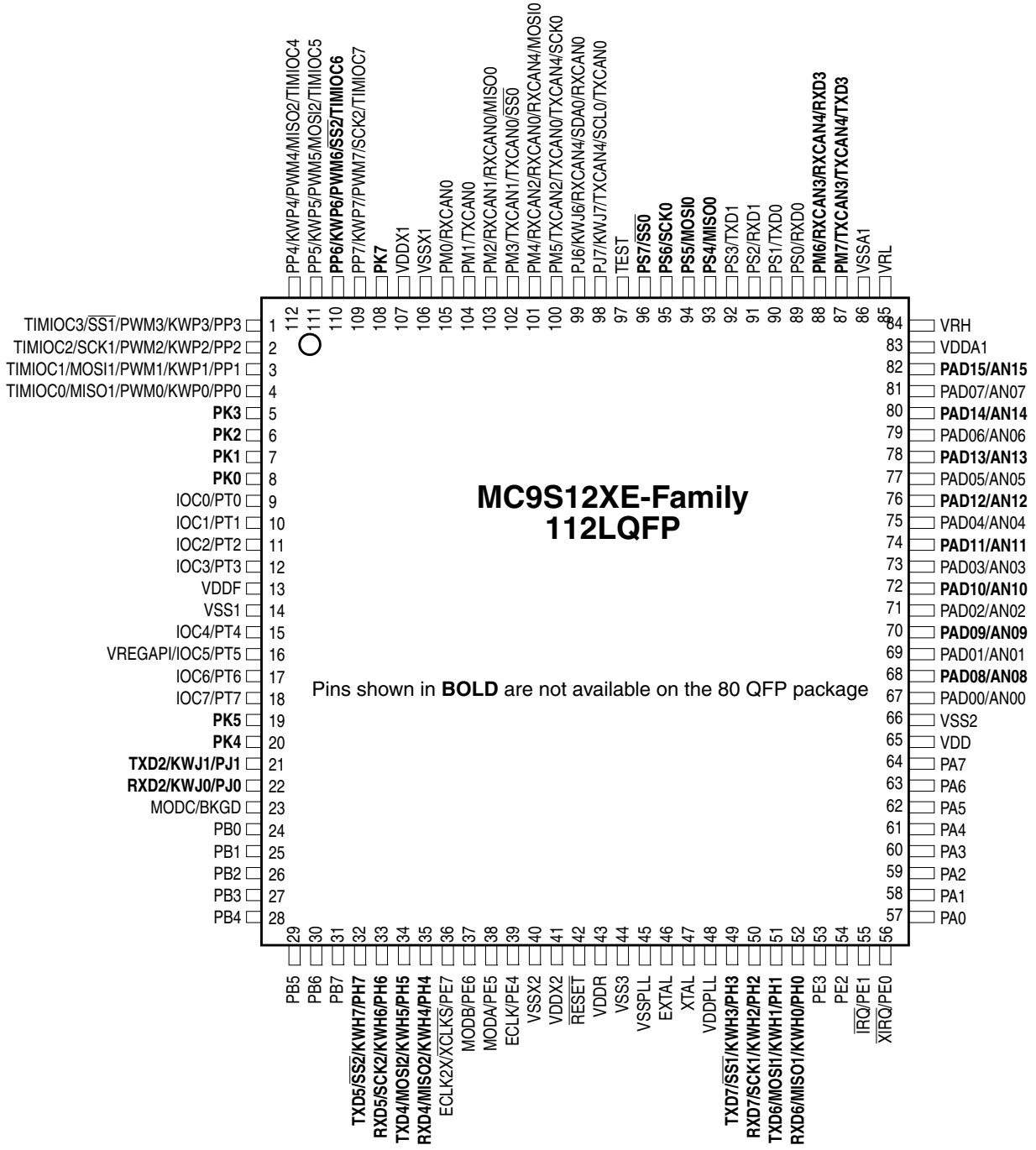



**Figure 1-6. MC9S12XE-Family Pin Assignments 112-pin LQFP Package**

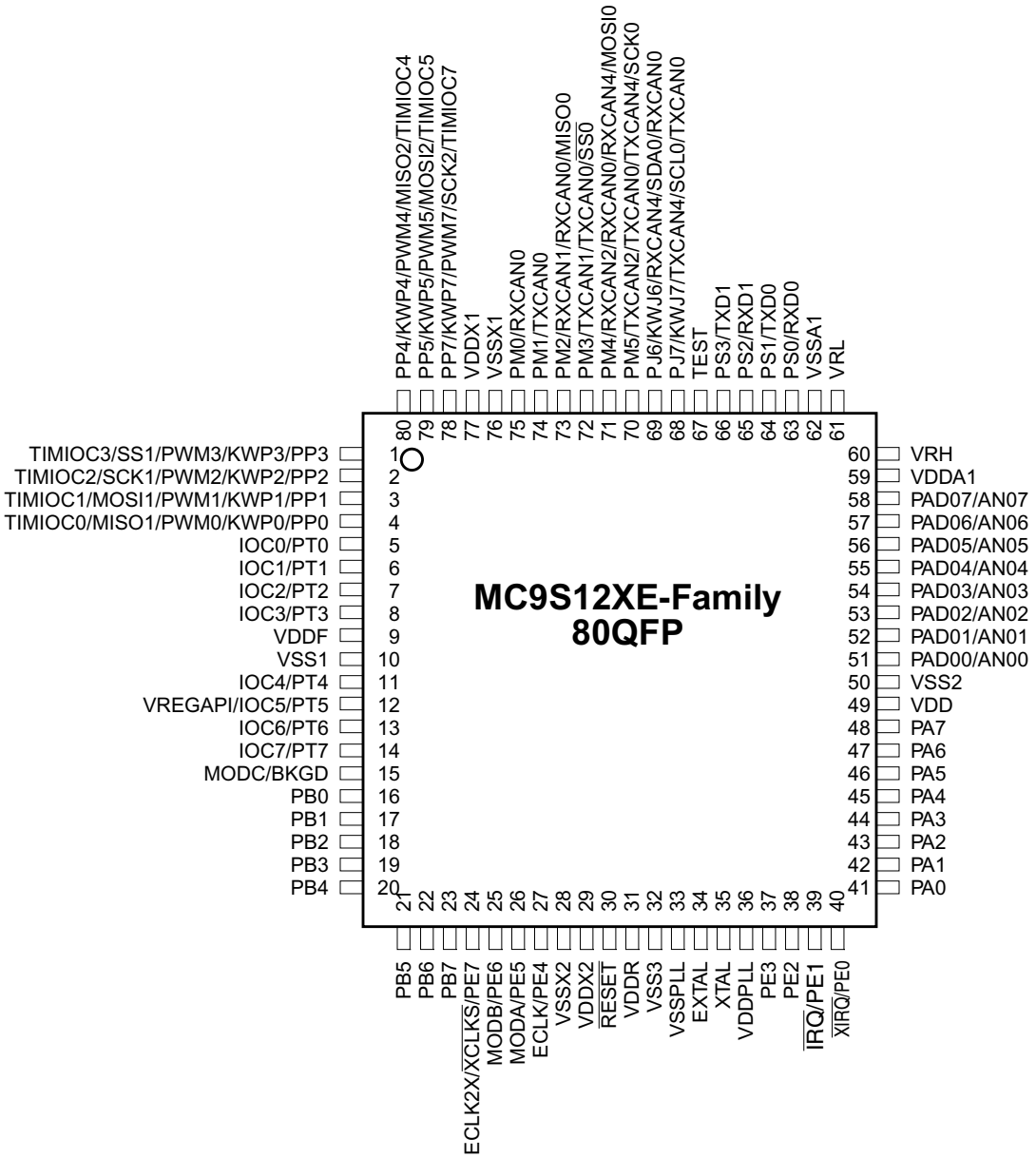



**Figure 1-7. MC9S12XE-Family Pin Assignments 80-pin QFP Package**

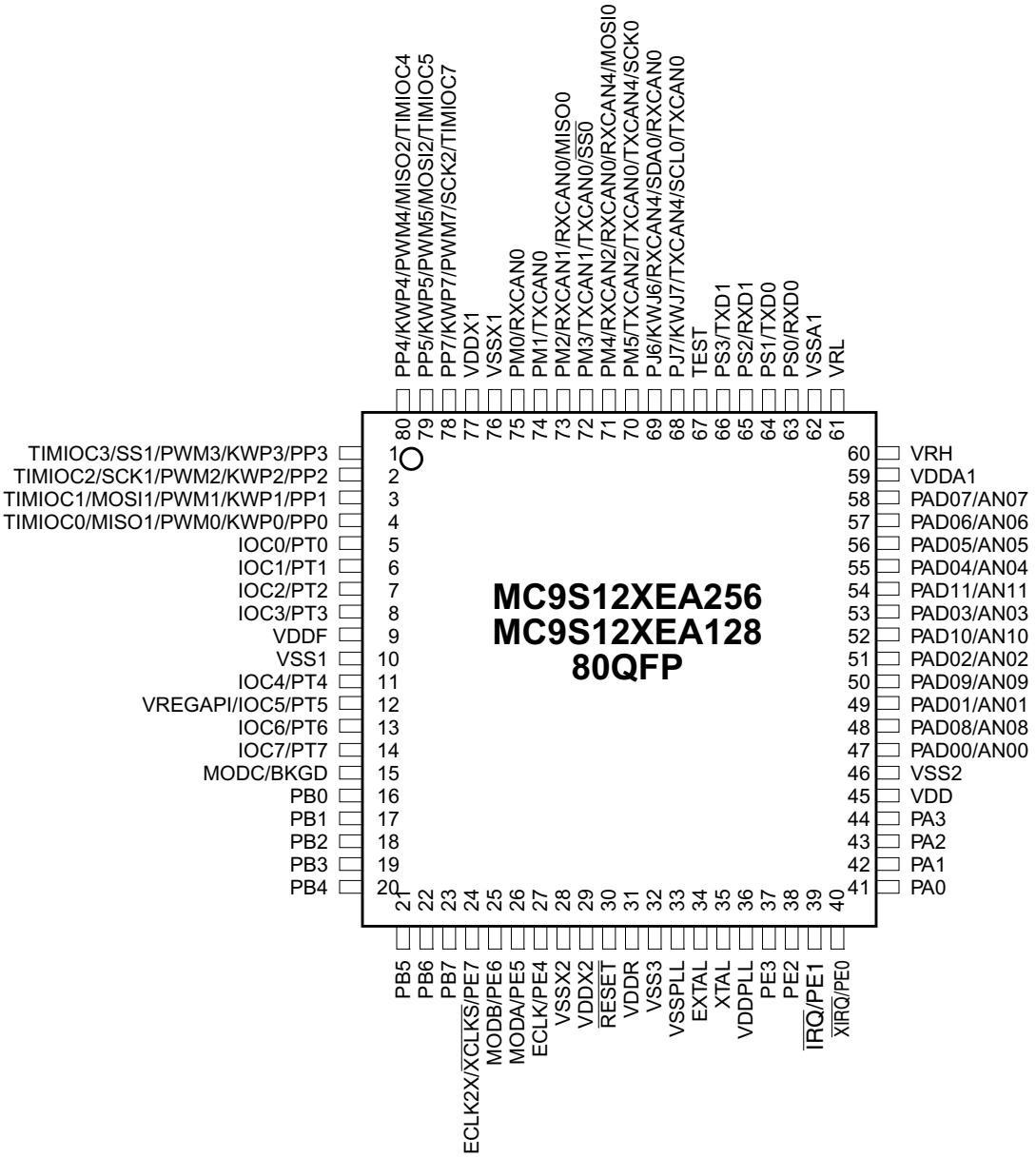



**Figure 1-8. MC9S12XEA256/MC9S12XEA128 80-pin QFP Package Pin Assignment**

**NOTE**

SPECIAL BOND-OUT TO PROVIDE ACCESS TO EXTRA ADC CHANNELS IN 80QFP. WARNING: NOT PIN-COMPATIBLE WITH REST OF FAMILY. THE MC9S12XET256 AND MC9S12XEG128 USE THE STANDARD 80QFP BOND-OUT, COMPATIBLE WITH OTHER FAMILY MEMBERS.

### 1.2.2 Pin Assignment Overview

Table 1-7 provides a summary of which Ports are available for each package option.

Routing of pin functions is summarized in Table 1-8.

Table 1-9 provides a pin out summary listing the availability of individual pins for each package option.

Table 1-10 provides a list of individual pin functionality

**Table 1-7. Port Availability by Package Option**

| Port | 208 MAPBGA | 144 LQFP | 112 LQFP | Standard 80 QFP | XEA256[1] 80 QFP |
|---|---|---|---|---|---|
| Port AD/ADC Channels | 32/32 | 24/24 | 16/16 | 8/8 | 12/12 |
| Port A pins | 8 | 8 | 8 | 8 | 4 |
| Port B pins | 8 | 8 | 8 | 8 | 8 |
| Port C pins | 8 | 8 | 0 | 0 | 0 |
| Port D pins | 8 | 8 | 0 | 0 | 0 |
| Port E pins inc. IRQ/XIRQ input only | 8 | 8 | 8 | 8 | 8 |
| Port F | 8 | 0 | 0 | 0 | 0 |
| Port H | 8 | 8 | 8 | 0 | 0 |
| Port J | 8 | 7 | 4 | 2 | 2 |
| Port K | 8 | 8 | 7 | 0 | 0 |
| Port L | 8 | 0 | 0 | 0 | 0 |
| Port M | 8 | 8 | 8 | 6 | 6 |
| Port P | 8 | 8 | 8 | 7 | 7 |
| Port R | 8 | 0 | 0 | 0 | 0 |
| Port S | 8 | 8 | 8 | 4 | 4 |
| Port T | 8 | 8 | 8 | 8 | 8 |
| **Sum of Ports** | **152** | **119** | **91** | **59** | **59** |
| I/O Power Pairs VDDX/VSSX | 7/7 | 4/4 | 2/2 | 2/2 | 2/2 |

1. The 9S12XEA256 is a special bondout for access to extra ADC channels in 80QFP.
Available in 80QFP / 256K memory size only. WARNING: NOT PIN-COMPATIBLE WITH REST OF FAMILY.
The 9S12XET256 is the standard 256K/80QFP bondout, compatible with other family members.

**Table 1-8. Peripheral - Port Routing Options[1]**

| | CAN0 | CAN1 | CAN2 | CAN3 | CAN4 | SCI0 | SCI1 | SCI2 | SCI3 | SCI4 | SCI5 | SCI6 | SCI7 | SPI0 | SPI1 | SPI2 | IIC0 | IIC1 | $\overline{CS0}$ | $\overline{CS1}$ | $\overline{CS2}$ | $\overline{CS3}$ | TIM |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| PF[0] | | | | | | | | | | | | | | | | | | | X | | | | |
| PF[1] | | | | | | | | | | | | | | | | | | | | X | | | |
| PF[2] | | | | | | | | | | | | | | | | | | | | | X | | |
| PF[3] | | | | | | | | | | | | | | | | | | | | | | X | |
| PF[5:4] | | | | | | | | | | | | | | | | | X | | | | | | |
| PF[7:6] | | | | | | | | | X | | | | | | | | | | | | | | |
| PH[1:0] | | | | | | | | | | | | O | | | X | | | | | | | | |
| PH[3:2] | | | | | | | | | | | | | O | | X | | | | | | | | |
| PH[5:4] | | | | | | | | | | O | | | | | | X | | | | | | | |
| PH[7:6] | | | | | | | | | | | O | | | | | X | | | | | | | |
| PJ[0] | | | | | | | | O | | | | | | | | | | | | | | O | |
| PJ[1] | | | | | | | | O | | | | | | | | | | | | | | | |
| PJ[2] | | | | | | | | | | | | | | | | | | | | O | | | |
| PJ[3] | | | | | | | | | | | | | | | | | | | | | | | |
| PJ[4] | | | | | | | | | | | | | | | | | | O | O | | | | |
| PJ[5] | | | | | | | | | | | | | | | | | | O | | | O | | |
| PJ[7:6] | X | | | | O | | | | | | | | | | | | O | | | | | | |
| PL[1:0] | | | | | | | | | | X | | | | | | | | | | | | | |
| PL[3:2] | | | | | | | | | | | X | | | | | | | | | | | | |
| PL[5:4] | | | | | | | | | | | | X | | | | | | | | | | | |
| PL[7:6] | | | | | | | | | | | | | X | | | | | | | | | | |
| PM[1:0] | O | | | | | | | | | | | | | | | | | | | | | | |
| PM[3:2] | X | O | | | | | | | | | | | | X | | | | | | | | | |
| PM[5:4] | X | | O | | X | | | | | | | | | X | | | | | | | | | |
| PM[7:6] | | | | O | X | | | | O | | | | | | | | | | | | | | |
| PP[3:0] | | | | | | | | | | | | | | | O | | | | | | | | X |
| PP[7:4] | | | | | | | | | | | | | | | | O | | | | | | | X |
| PR[7:0] | | | | | | | | | | | | | | | | | | | | | | | O |

**Table 1-8. Peripheral - Port Routing Options[1] (continued)**

| | CAN0 | CAN1 | CAN2 | CAN3 | CAN4 | SCI0 | SCI1 | SCI2 | SCI3 | SCI4 | SCI5 | SCI6 | SCI7 | SPI0 | SPI1 | SPI2 | IIC0 | IIC1 | $\overline{CS0}$ | $\overline{CS1}$ | $\overline{CS2}$ | $\overline{CS3}$ | TIM |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| PS[1:0] | | | | | | O | | | | | | | | | | | | | | | | | |
| PS[3:2] | | | | | | | O | | | | | | | | | | | | | | | | |
| PS[7:4] | | | | | | | | | | | | | | O | | | | | | | | | |

1. “O” denotes reset condition, “X” denotes a possible rerouting under software control

**Table 1-9. Pin-Out Summary (Sheet 1 of 7)**

| 208 MAPBGA | LQFP 144 | LQFP 112 | QFP[1] 80 | Pin | 2nd Func. | 3rd Func. | 4th Func. | 5th Func. |
|---|---|---|---|---|---|---|---|---|
| D4 | 1 | 1 | 1 | PP3 | KWP3 | PWM3 | $\overline{SS1}$ | TIMIOC3 |
| B2 | 2 | 2 | 2 | PP2 | KWP2 | PWM2 | SCK1 | TIMIOC2 |
| C2 | 3 | 3 | 3 | PP1 | KWP1 | PWM1 | MOSI1 | TIMIOC1 |
| D3 | 4 | 4 | 4 | PP0 | KWP0 | PWM0 | MISO1 | TIMIOC0 |
| D2 | | | | PJ3 | KWJ3 | | | |
| C1 | 5 | | | PJ2 | KWJ2 | $\overline{CS1}$ | | |
| E4 | 6 | | | PK6 | ADDR22 | ACC2 | | |
| E2 | 7 | 5 | | PK3 | ADDR19 | IQSTAT3 | | |
| E3 | 8 | 6 | | PK2 | ADDR18 | IQSTAT2 | | |
| D1 | 9 | 7 | | PK1 | ADDR17 | IQSTAT1 | | |
| E1 | 10 | 8 | | PK0 | ADDR16 | IQSTAT0 | | |
| VDDX | | | | VDDX7 | | | | |
| VSSX | | | | VSSX7 | | | | |
| F3 | 11 | 9 | 5 | PT0 | IOC0 | | | |
| F2 | | | | PR0 | TIMIOC0 | | | |
| G4 | 12 | 10 | 6 | PT1 | IOC1 | | | |
| F1 | | | | PR1 | TIMIOC1 | | | |
| G1 | 13 | 11 | 7 | PT2 | IOC2 | | | |
| G3 | | | | PR2 | TIMIOC2 | | | |
| G2 | 14 | 12 | 8 | PT3 | IOC3 | | | |
| H1 | | | | PR3 | TIMIOC3 | | | |
| H4 | 15 | 13 | 9 | VDDF | | | | |

**Table 1-9. Pin-Out Summary (Sheet 2 of 7)**

| 208 MAPBGA | LQFP 144 | LQFP 112 | QFP[1] 80 | Pin | 2nd Func. | 3rd Func. | 4th Func. | 5th Func. |
|---|---|---|---|---|---|---|---|---|
| J4 | 16 | 14 | 10 | VSS1 | | | | |
| H3 | 17 | 15 | 11 | PT4 | IOC4 | | | |
| H2 | | | | PR4 | TIMIOC4 | | | |
| J1 | 18 | 16 | 12 | PT5 | IOC5 | VREGAPI | | |
| J2 | | | | PR5 | TIMIOC5 | | | |
| J3 | 19 | 17 | 13 | PT6 | IOC6 | | | |
| K1 | | | | PR6 | TIMIOC6 | | | |
| K2 | 20 | 18 | 14 | PT7 | IOC7 | | | |
| K4 | | | | PR7 | TIMIOC7 | | | |
| L1 | 21 | 19 | | PK5 | ADDR21 | ACC1 | | |
| K3 | 22 | 20 | | PK4 | ADDR20 | ACC0 | | |
| L2 | 23 | 21 | | PJ1 | KWJ1 | TXD2 | | |
| M1 | 24 | 22 | | PJ0 | KWJ0 | RXD2 | $\overline{CS3}$ | |
| L3 | 25 | 23 | 15 | BKGD | MODC | | | |
| VDDX | 26 | | | VDDX4 | | | | |
| VSSX | 27 | | | VSSX4 | | | | |
| M2 | 28 | | | PC0 | DATA8 | | | |
| M4 | 29 | | | PC1 | DATA9 | | | |
| N1 | 30 | | | PC2 | DATA10 | | | |
| N2 | 31 | | | PC3 | DATA11 | | | |
| P1 | 32 | 24 | 16 | PB0 | ADDR0 | IVD0 | $\overline{UDS}$ | |
| M3 | 33 | 25 | 17 | PB1 | ADDR1 | IVD1 | | |
| N3 | 34 | 26 | 18 | PB2 | ADDR2 | IVD2 | | |
| P2 | 35 | 27 | 19 | PB3 | ADDR3 | IVD3 | | |
| P3 | 36 | 28 | 20 | PB4 | ADDR4 | IVD4 | | |
| R2 | 37 | 29 | 21 | PB5 | ADDR5 | IVD5 | | |
| R3 | 38 | 30 | 22 | PB6 | ADDR6 | IVD6 | | |
| R4 | 39 | 31 | 23 | PB7 | ADDR7 | IVD7 | | |
| P4 | 40 | | | PC4 | DATA12 | | | |

**Table 1-9. Pin-Out Summary (Sheet 3 of 7)**

| 208 MAPBGA | LQFP 144 | LQFP 112 | QFP[1] 80 | Pin | 2nd Func. | 3rd Func. | 4th Func. | 5th Func. |
|---|---|---|---|---|---|---|---|---|
| T3 | 41 | | | PC5 | DATA13 | | | |
| R5 | 42 | | | PC6 | DATA14 | | | |
| N4 | 43 | | | PC7 | DATA15 | | | |
| T4 | | | | PL3 | TXD5 | | | |
| T5 | 44 | 32 | | PH7 | KWH7 | $\overline{SS2}$ | TXD5 | |
| P5 | | | | PL2 | RXD5 | | | |
| R6 | 45 | 33 | | PH6 | KWH6 | SCK2 | RXD5 | |
| N5 | | | | PL1 | TXD4 | | | |
| T6 | 46 | 34 | | PH5 | KWH5 | MOSI2 | TXD4 | |
| P6 | | | | PL0 | RXD4 | | | |
| R7 | 47 | 35 | | PH4 | KWH4 | MISO2 | RXD4 | |
| T7 | 48 | 36 | 24 | PE7 | $\overline{XCLKS}$ | ECLKX2 | | |
| N6 | 49 | 37 | 25 | PE6 | MODB | $\overline{TAGHI}$ | | |
| R8 | 50 | 38 | 26 | PE5 | MODA | $\overline{TAGLO}$ | $\overline{RE}$ | |
| P7 | 51 | 39 | 27 | PE4 | ECLK | | | |
| VSSX | 52 | 40 | 28 | VSSX2 | | | | |
| VDDX | 53 | 41 | 29 | VDDX2 | | | | |
| P8 | 54 | 42 | 30 | $\overline{RESET}$ | | | | |
| N8 | 55 | 43 | 31 | VDDR | | | | |
| N9 | 56 | 44 | 32 | VSS3 | | | | |
| R9/T8 | 57 | 45 | 33 | VSSPLL | | | | |
| T9 | 58 | 46 | 34 | EXTAL | | | | |
| T10 | 59 | 47 | 35 | XTAL | | | | |
| R10/T11 | 60 | 48 | 36 | VDDPLL | | | | |
| P9 | | | | PL7 | TXD7 | | | |
| N10 | 61 | 49 | | PH3 | KWH3 | $\overline{SS1}$ | TXD7 | |
| P10 | | | | PL6 | RXD7 | | | |
| R11 | 62 | 50 | | PH2 | KWH2 | SCK1 | RXD7 | |
| T12 | | | | PL5 | TXD6 | | | |

**Table 1-9. Pin-Out Summary (Sheet 4 of 7)**

| 208 MAPBGA | LQFP 144 | LQFP 112 | QFP[1] 80 | Pin | 2nd Func. | 3rd Func. | 4th Func. | 5th Func. |
|---|---|---|---|---|---|---|---|---|
| N11 | 63 | 51 | | PH1 | KWH1 | MOSI1 | TXD6 | |
| R12 | | | | PL4 | RXD6 | | | |
| P11 | 64 | 52 | | PH0 | KWH0 | MISO1 | RXD6 | |
| T13 | 65 | | | PD0 | DATA0 | | | |
| R13 | 66 | | | PD1 | DATA1 | | | |
| T14 | 67 | | | PD2 | DATA2 | | | |
| R14 | 68 | | | PD3 | DATA3 | | | |
| VDDX | | | | VDDX5 | | | | |
| VSSX | | | | VSSX5 | | | | |
| R15 | 69 | 53 | 37 | PE3 | $\overline{\text{LSTRB}}$ | $\overline{\text{LDS}}$ | EROMCTL | |
| P12 | 70 | 54 | 38 | PE2 | R$\overline{\text{W}}$ | $\overline{\text{WE}}$ | | |
| N13 | 71 | 55 | 39 | PE1 | $\overline{\text{IRQ}}$ | | | |
| P13 | 72 | 56 | 40 | PE0 | $\overline{\text{XIRQ}}$ | | | |
| P14 | 73 | 57 | 41 | PA0 | ADDR8 | IVD8 | | |
| N14 | 74 | 58 | 42 | PA1 | ADDR9 | IVD9 | | |
| M14 | 75 | 59 | 43 | PA2 | ADDR10 | IVD10 | | |
| P15 | 76 | 60 | 44 | PA3 | ADDR11 | IVD11 | | |
| P16 | 77 | 61 | 45 | PA4 | ADDR12 | IVD12 | | |
| N15 | 78 | 62 | 46 | PA5 | ADDR13 | IVD13 | | |
| M13 | 79 | 63 | 47 | PA6 | ADDR14 | IVD14 | | |
| N16 | 80 | 64 | 48 | PA7 | ADDR15 | IVD15 | | |
| VSSX | 81 | | | VSSX3 | | | | |
| VDDX | 82 | | | VDDX3 | | | | |
| L14 | 83 | | | PD4 | DATA4 | | | |
| M15 | 84 | | | PD5 | DATA5 | | | |
| M16 | 85 | | | PD6 | DATA6 | | | |
| K14 | 86 | | | PD7 | DATA7 | | | |
| K13 | 87 | 65 | 49 | VDD | | | | |
| J13 | 88 | 66 | 50 | VSS2 | | | | |

**Table 1-9. Pin-Out Summary (Sheet 5 of 7)**

| 208 MAPBGA | LQFP 144 | LQFP 112 | QFP[1] 80 | Pin | 2nd Func. | 3rd Func. | 4th Func. | 5th Func. |
|---|---|---|---|---|---|---|---|---|
| L15 | 89 | 67 | 51 | PAD00 | AN00 | | | |
| L16 | 90 | 68 | | PAD08 | AN08 | | | |
| K15 | | | | PAD24 | AN24 | | | |
| K16 | 91 | 69 | 52 | PAD01 | AN01 | | | |
| J14 | 92 | 70 | | PAD09 | AN09 | | | |
| J15 | | | | PAD25 | AN25 | | | |
| J16 | 93 | 71 | 53 | PAD02 | AN02 | | | |
| H16 | 94 | 72 | | PAD10 | AN10 | | | |
| H14 | | | | PAD26 | AN26 | | | |
| H13 | | | | VSSA2 | | | | |
| G13 | | | | VDDA2 | | | | |
| H15 | 95 | 73 | 54 | PAD03 | AN03 | | | |
| G16 | 96 | 74 | | PAD11 | AN11 | | | |
| F16 | | | | PAD27 | AN27 | | | |
| G15 | 97 | 75 | 55 | PAD04 | AN04 | | | |
| G14 | 98 | 76 | | PAD12 | AN12 | | | |
| E16 | | | | PAD28 | AN28 | | | |
| F14 | 99 | 77 | 56 | PAD05 | AN05 | | | |
| F15 | 100 | 78 | | PAD13 | AN13 | | | |
| D16 | | | | PAD29 | AN29 | | | |
| E15 | 101 | 79 | 57 | PAD06 | AN06 | | | |
| C16 | 102 | 80 | | PAD14 | AN14 | | | |
| D15 | | | | PAD30 | AN30 | | | |
| C15 | 103 | 81 | 58 | PAD07 | AN07 | | | |
| E14 | 104 | 82 | | PAD15 | AN15 | | | |
| B15 | | | | PAD31 | AN31 | | | |
| C14 | 105 | | | PAD16 | AN16 | | | |
| D14 | 106 | | | PAD17 | AN17 | | | |
| F13 | 107 | 83 | 59 | VDDA1 | | | | |

**Table 1-9. Pin-Out Summary (Sheet 6 of 7)**

| 208 MAPBGA | LQFP 144 | LQFP 112 | QFP[1] 80 | Pin | 2nd Func. | 3rd Func. | 4th Func. | 5th Func. |
|---|---|---|---|---|---|---|---|---|
| D13 | 108 | 84 | 60 | VRH | | | | |
| C13 | 109 | 85 | 61 | VRL | | | | |
| E13 | 110 | 86 | 62 | VSSA1 | | | | |
| B14 | 111 | | | PAD18 | AN18 | | | |
| A14 | 112 | | | PAD19 | AN19 | | | |
| C12 | 113 | | | PAD20 | AN20 | | | |
| B13 | 114 | | | PAD21 | AN21 | | | |
| D12 | 115 | | | PAD22 | AN22 | | | |
| B12 | 116 | | | PAD23 | AN23 | | | |
| C11 | 117 | 87 | | PM7 | TXCAN3 | TXCAN4 | TXD3 | |
| A13 | 118 | 88 | | PM6 | RXCAN3 | RXCAN4 | RXD3 | |
| D11 | 119 | 89 | 63 | PS0 | RXD0 | | | |
| B11 | 120 | 90 | 64 | PS1 | TXD0 | | | |
| C10 | 121 | 91 | 65 | PS2 | RXD1 | | | |
| A12 | 122 | 92 | 66 | PS3 | TXD1 | | | |
| VSSX | | | | VSSX6 | | | | |
| VDDX | | | | VDDX6 | | | | |
| B10 | 123 | 93 | | PS4 | MISO0 | | | |
| A11 | 124 | 94 | | PS5 | MOSI0 | | | |
| A10 | 125 | 95 | | PS6 | SCK0 | | | |
| C9 | 126 | 96 | | PS7 | $\overline{SS0}$ | | | |
| B9 | 127 | 97 | 67 | TEST | | | | |
| D9 | 128 | 98 | 68 | PJ7 | KWJ7 | TXCAN4 | SCL0 | TXCAN0 |
| A9 | 129 | 99 | 69 | PJ6 | KWJ6 | RXCAN4 | SDA0 | RXCAN0 |
| C8 | 130 | | | PJ5 | KWJ5 | SCL1 | $\overline{CS2}$ | |
| B8 | | | | PF0 | $\overline{CS0}$ | | | |
| D8 | 131 | | | PJ4 | KWJ4 | SDA1 | $\overline{CS0}$ | |
| A8 | | | | PF1 | $\overline{CS1}$ | | | |
| D7 | 132 | 100 | 70 | PM5 | TXCAN2 | TXCAN0 | TXCAN4 | SCK0 |

**Table 1-9. Pin-Out Summary (Sheet 7 of 7)**

| 208 MAPBGA | LQFP 144 | LQFP 112 | QFP(1) 80 | Pin | 2nd Func. | 3rd Func. | 4th Func. | 5th Func. |
|---|---|---|---|---|---|---|---|---|
| B7 | | | | PF2 | $\overline{CS2}$ | | | |
| C7 | 133 | 101 | 71 | PM4 | RXCAN2 | RXCAN0 | RXCAN4 | MOSI0 |
| A7 | | | | PF3 | $\overline{CS3}$ | | | |
| D6 | 134 | 102 | 72 | PM3 | TXCAN1 | TXCAN0 | $\overline{SS0}$ | |
| B6 | | | | PF4 | SDA0 | | | |
| C6 | 135 | 103 | 73 | PM2 | RXCAN1 | RXCAN0 | MISO0 | |
| A6 | | | | PF5 | SCL0 | | | |
| A5 | 136 | 104 | 74 | PM1 | TXCAN0 | | | |
| B5 | | | | PF6 | RXD3 | | | |
| A4 | 137 | 105 | 75 | PM0 | RXCAN0 | | | |
| B4 | | | | PF7 | TXD3 | | | |
| VSSX | 138 | 106 | 76 | VSSX1 | | | | |
| VDDX | 139 | 107 | 77 | VDDX1 | | | | |
| C5 | 140 | 108 | | PK7 | ROMCTL | $\overline{EWAIT}$ | | |
| A3 | 141 | 109 | 78 | PP7 | KWP7 | PWM7 | SCK2 | TIMIOC7 |
| B3 | 142 | 110 | | PP6 | KWP6 | PWM6 | $\overline{SS2}$ | TIMIOC6 |
| C4 | 143 | 111 | 79 | PP5 | KWP5 | PWM5 | MOSI2 | TIMIOC5 |
| C3 | 144 | 112 | 80 | PP4 | KWP4 | PWM4 | MISO2 | TIMIOC4 |

1. Standard 80QFP only. NOTE that XEA256 80QFP is not compatible

**Table 1-10. Signal Properties Summary (Sheet 1 of 4)**

| Pin Name Function 1 | Pin Name Function 2 | Pin Name Function 3 | Pin Name Function 4 | Pin Name Function 5 | Power Supply | Internal Pull Resistor | | Description |
|---|---|---|---|---|---|---|---|---|
| | | | | | | CTRL | Reset State | |
| EXTAL | — | — | — | — | $V_{DDPLL}$ | NA | NA | Oscillator pins |
| XTAL | — | — | — | — | $V_{DDPLL}$ | NA | NA | |
| $\overline{\text{RESET}}$ | — | — | — | — | $V_{DDX}$ | PULLUP | | External reset |
| TEST | — | — | — | — | N.A. | $\overline{\text{RESET}}$ pin | DOWN | Test input |
| BKGD | MODC | — | — | — | $V_{DDX}$ | Always on | Up | Background debug |
| PAD[31:16] | AN[31:16] | — | — | — | $V_{DDA}$ | PER0AD1 PER1AD1 | Disabled | Port AD inputs of ATD1, analog inputs of ATD1 |
| PAD[15:0] | AN[15:0] | — | — | — | $V_{DDA}$ | PER0AD0 PER1AD0 | Disabled | Port AD inputs of ATD0, analog inputs of ATD0 |
| PA[7:0] | ADDR[15:8] | IVD[15:8] | — | — | $V_{DDX}$ | PUCR | Disabled | Port A I/O, address bus, internal visibility data |
| PB[7:1] | ADDR[7:1] | IVD[7:0] | — | — | $V_{DDX}$ | PUCR | Disabled | Port B I/O, address bus, internal visibility data |
| PB0 | ADDR0 | $\overline{\text{UDS}}$ | | | $V_{DDX}$ | PUCR | Disabled | Port B I/O, address bus, upper data strobe |
| PC[7:0] | DATA[15:8] | — | — | — | $V_{DDX}$ | PUCR | Disabled | Port C I/O, data bus |
| PD[7:0] | DATA[7:0] | — | — | — | $V_{DDX}$ | PUCR | Disabled | Port D I/O, data bus |
| PE7 | ECLKX2 | $\overline{\text{XCLKS}}$ | — | — | $V_{DDX}$ | PUCR | Up | Port E I/O, system clock output, clock select |
| PE6 | $\overline{\text{TAGHI}}$ | MODB | — | — | $V_{DDX}$ | While $\overline{\text{RESET}}$ pin is low: down | | Port E I/O, tag high, mode input |
| PE5 | $\overline{\text{RE}}$ | MODA | $\overline{\text{TAGLO}}$ | — | $V_{DDX}$ | While $\overline{\text{RESET}}$ pin is low: down | | Port E I/O, read enable, mode input, tag low input |
| PE4 | ECLK | — | — | — | $V_{DDX}$ | PUCR | Up | Port E I/O, bus clock output |
| PE3 | $\overline{\text{LSTRB}}$ | $\overline{\text{LDS}}$ | EROMCTL | — | $V_{DDX}$ | PUCR | Up | Port E I/O, low byte data strobe, EROMON control |
| PE2 | R/$\overline{\text{W}}$ | $\overline{\text{WE}}$ | — | — | $V_{DDX}$ | PUCR | Up | Port E I/O, read/write |
| PE1 | $\overline{\text{IRQ}}$ | — | — | — | $V_{DDX}$ | PUCR | Up | Port E Input, maskable interrupt |
| PE0 | $\overline{\text{XIRQ}}$ | — | — | — | $V_{DDX}$ | PUCR | Up | Port E input, non-maskable interrupt |
| PF7 | TXD3 | — | — | — | $V_{DDX}$ | PERF/ PPSF | Up | Port F I/O, interrupt, TXD of SCI3 |
| PF6 | RXD3 | — | — | — | $V_{DDX}$ | PERF/ PPSF | Up | Port F I/O, interrupt, RXD of SCI3 |
| PF5 | SCL0 | — | — | — | $V_{DDX}$ | PERF/ PPSF | Up | Port F I/O, interrupt, SCL of IIC0 |
| PF4 | SDA0 | — | — | — | $V_{DDX}$ | PERF/ PPSF | Up | Port F I/O, interrupt, SDA of IIC0 |
| PF3 | $\overline{\text{CS3}}$ | — | — | — | $V_{DDX}$ | PERF/ PPSF | Up | Port F I/O, interrupt, chip select 3 |

**Table 1-10. Signal Properties Summary (Sheet 2 of 4)**

| Pin Name Function 1 | Pin Name Function 2 | Pin Name Function 3 | Pin Name Function 4 | Pin Name Function 5 | Power Supply | Internal Pull Resistor | | Description |
|---|---|---|---|---|---|---|---|---|
| | | | | | | CTRL | Reset State | |
| PF2 | $\overline{CS2}$ | — | — | — | $V_{DDX}$ | PERF/ PPSF | Up | Port F I/O, interrupt, chip select 2 |
| PF1 | $\overline{CS1}$ | — | — | — | $V_{DDX}$ | PERF/ PPSF | Up | Port F I/O, interrupt, chip select 1 |
| PF0 | $\overline{CS0}$ | — | — | — | $V_{DDX}$ | PERF/ PPSF | Up | Port F I/O, interrupt, chip select 0 |
| PH7 | KWH7 | $\overline{SS2}$ | TXD5 | — | $V_{DDX}$ | PERH/ PPSH | Disabled | Port H I/O, interrupt, $\overline{SS}$ of SPI2, TXD of SCI5 |
| PH6 | KWH6 | SCK2 | RXD5 | — | $V_{DDX}$ | PERH/ PPSH | Disabled | Port H I/O, interrupt, SCK of SPI2, RXD of SCI5 |
| PH5 | KWH5 | MOSI2 | TXD4 | — | $V_{DDX}$ | PERH/ PPSH | Disabled | Port H I/O, interrupt, MOSI of SPI2, TXD of SCI4 |
| PH4 | KWH4 | MISO2 | RXD4 | — | $V_{DDX}$ | PERH/PPSH | Disabled | Port H I/O, interrupt, MISO of SPI2, RXD of SCI4 |
| PH3 | KWH3 | $\overline{SS1}$ | TXD7 | — | $V_{DDX}$ | PERH/PPSH | Disabled | Port H I/O, interrupt, $\overline{SS}$ of SPI1 |
| PH2 | KWH2 | SCK1 | RXD7 | — | $V_{DDX}$ | PERH/PPSH | Disabled | Port H I/O, interrupt, SCK of SPI1 |
| PH1 | KWH1 | MOSI1 | TXD6 | — | $V_{DDX}$ | PERH/PPSH | Disabled | Port H I/O, interrupt, MOSI of SPI1 |
| PH0 | KWH0 | MISO1 | RXD6 | — | $V_{DDX}$ | PERH/PPSH | Disabled | Port H I/O, interrupt, MISO of SPI1 |
| PJ7 | KWJ7 | TXCAN4 | SCL0 | TXCAN0 | $V_{DDX}$ | PERJ/ PPSJ | Up | Port J I/O, interrupt, TX of CAN4, SCL of IIC0, TX of CAN0 |
| PJ6 | KWJ6 | RXCAN4 | SDA0 | RXCAN0 | $V_{DDX}$ | PERJ/ PPSJ | Up | Port J I/O, interrupt, RX of CAN4, SDA of IIC0, RX of CAN0 |
| PJ5 | KWJ5 | SCL1 | $\overline{CS2}$ | — | $V_{DDX}$ | PERJ/ PPSJ | Up | Port J I/O, interrupt, SCL of IIC1, chip select 2 |
| PJ4 | KWJ4 | SDA1 | $\overline{CS0}$ | — | $V_{DDX}$ | PERJ/ PPSJ | Up | Port J I/O, interrupt, SDA of IIC1, chip select 0 |
| PJ3 | KWJ3 | — | — | — | $V_{DDX}$ | PERJ/ PPSJ | Up | Port J I/O, interrupt, |
| PJ2 | KWJ2 | $\overline{CS1}$ | — | — | $V_{DDX}$ | PERJ/ PPSJ | Up | Port J I/O, interrupt, chip select 1 |
| PJ1 | KWJ1 | TXD2 | — | — | $V_{DDX}$ | PERJ/ PPSJ | Up | Port J I/O, interrupt, TXD of SCI2 |
| PJ0 | KWJ0 | RXD2 | $\overline{CS3}$ | — | $V_{DDX}$ | PERJ/ PPSJ | Up | Port J I/O, interrupt, RXD of SCI2 |
| PK7 | $\overline{EWAIT}$ | ROMCTL | — | — | $V_{DDX}$ | PUCR | Up | Port K I/O, EWAIT input, ROM on control |

**Table 1-10. Signal Properties Summary (Sheet 3 of 4)**

| Pin Name Function 1 | Pin Name Function 2 | Pin Name Function 3 | Pin Name Function 4 | Pin Name Function 5 | Power Supply | Internal Pull Resistor | | Description |
|---|---|---|---|---|---|---|---|---|
| | | | | | | CTRL | Reset State | |
| PK[6:4] | ADDR [22:20] | ACC[2:0] | — | — | $V_{DDX}$ | PUCR | Up | Port K I/O, extended addresses, access source for external access |
| PK[3:0] | ADDR [19:16] | IQSTAT [3:0] | — | — | $V_{DDX}$ | PUCR | Up | Extended address, PIPE status |
| PL7 | TXD7 | — | — | — | $V_{DDX}$ | PERL/ PPSL | Up | Port L I/O, TXD of SCI7 |
| PL6 | RXD7 | — | — | — | $V_{DDX}$ | PERL/ PPSL | Up | Port LI/O, RXD of SCI7 |
| PL5 | TXD6 | — | — | — | $V_{DDX}$ | PERL/ PPSL | Up | Port L I/O, TXD of SCI6 |
| PL4 | RXD6 | — | — | — | $V_{DDX}$ | PERL/ PPSL | Up | Port LI/O, RXD of SCI6 |
| PL3 | TXD5 | — | — | — | $V_{DDX}$ | PERL/ PPSL | Up | Port L I/O, TXD of SCI5 |
| PL2 | RXD5 | — | — | — | $V_{DDX}$ | PERL/ PPSL | Up | Port LI/O, RXD of SCI5 |
| PL1 | TXD4 | — | — | — | $V_{DDX}$ | PERL/ PPSL | Up | Port L I/O, TXD of SCI4 |
| PL0 | RXD4 | — | — | — | $V_{DDX}$ | PERL/ PPSL | Up | Port LI/O, RXD of SCI4 |
| PM7 | TXCAN3 | TXD3 | TXCAN4 | — | $V_{DDX}$ | PERM/ PPSM | Disabled | Port M I/O, TX of CAN3 and CAN4, TXD of SCI3 |
| PM6 | RXCAN3 | RXD3 | RXCAN4 | — | $V_{DDX}$ | PERM/PPSM | Disabled | Port M I/O RX of CAN3 and CAN4, RXD of SCI3 |
| PM5 | TXCAN2 | TXCAN0 | TXCAN4 | SCK0 | $V_{DDX}$ | PERM/PPSM | Disabled | Port M I/OCAN0, CAN2, CAN4, SCK of SPI0 |
| PM4 | RXCAN2 | RXCAN0 | RXCAN4 | MOSI0 | $V_{DDX}$ | PERM/PPSM | Disabled | Port M I/O, CAN0, CAN2, CAN4, MOSI of SPI0 |
| PM3 | TXCAN1 | TXCAN0 | $\overline{SS0}$ | — | $V_{DDX}$ | PERM/PPSM | Disabled | Port M I/O TX of CAN1, CAN0, $\overline{SS}$ of SPI0 |
| PM2 | RXCAN1 | RXCAN0 | MISO0 | — | $V_{DDX}$ | PERM/PPSM | Disabled | Port M I/O, RX of CAN1, CAN0, MISO of SPI0 |
| PM1 | TXCAN0 | | — | — | $V_{DDX}$ | PERM/PPSM | Disabled | Port M I/O, TX of CAN0 |
| PM0 | RXCAN0 | | — | — | $V_{DDX}$ | PERM/PPSM | Disabled | Port M I/O, RX of CAN0 |
| PP7 | KWP7 | PWM7 | SCK2 | TIMIOC7 | $V_{DDX}$ | PERP/ PPSP | Disabled | Port P I/O, interrupt, channel 7 of PWM/TIM , SCK of SPI2 |
| PP6 | KWP6 | PWM6 | $\overline{SS2}$ | TIMIOC6 | $V_{DDX}$ | PERP/ PPSP | Disabled | Port P I/O, interrupt, channel 6 of PWM/TIM, $\overline{SS}$ of SPI2 |
| PP5 | KWP5 | PWM5 | MOSI2 | TIMIOC5 | $V_{DDX}$ | PERP/ PPSP | Disabled | Port P I/O, interrupt, channel 5 of PWM/TIM, MOSI of SPI2 |

**Table 1-10. Signal Properties Summary (Sheet 4 of 4)**

| Pin Name Function 1 | Pin Name Function 2 | Pin Name Function 3 | Pin Name Function 4 | Pin Name Function 5 | Power Supply | Internal Pull Resistor | | Description |
|---|---|---|---|---|---|---|---|---|
| | | | | | | CTRL | Reset State | |
| PP4 | KWP4 | PWM4 | MISO2 | TIMIOC4 | $V_{DDX}$ | PERP/ PPSP | Disabled | Port P I/O, interrupt, channel 4 of PWM/TIM, MISO2 of SPI2 |
| PP3 | KWP3 | PWM3 | $\overline{SS1}$ | TIMIOC3 | $V_{DDX}$ | PERP/ PPSP | Disabled | Port P I/O, interrupt, channel 3 of PWM/TIM, $\overline{SS}$ of SPI1 |
| PP2 | KWP2 | PWM2 | SCK1 | TIMIOC2 | $V_{DDX}$ | PERP/ PPSP | Disabled | Port P I/O, interrupt, channel 2 of PWM/TIM, SCK of SPI1 |
| PP1 | KWP1 | PWM1 | MOSI1 | TIMIOC1 | $V_{DDX}$ | PERP/ PPSP | Disabled | Port P I/O, interrupt, channel 1 of PWM/TIM, MOSI of SPI1 |
| PP0 | KWP0 | PWM0 | MISO1 | TIMIOC0 | $V_{DDX}$ | PERP/ PPSP | Disabled | Port P I/O, interrupt, channel 0 of PWM/TIM, MISO2 of SPI1 |
| PR[7:0] | TIMIOC [7:0] | — | — | — | $V_{DDX}$ | PERR/ PPSR | Disabled | Port RI/O, TIM channels |
| PS7 | $\overline{SS0}$ | — | — | — | $V_{DDX}$ | PERS/ PPSS | Up | Port S I/O, $\overline{SS}$ of SPI0 |
| PS6 | SCK0 | — | — | — | $V_{DDX}$ | PERS/ PPSS | Up | Port S I/O, SCK of SPI0 |
| PS5 | MOSI0 | — | — | — | $V_{DDX}$ | PERS/ PPSS | Up | Port S I/O, MOSI of SPI0 |
| PS4 | MISO0 | — | — | — | $V_{DDX}$ | PERS/ PPSS | Up | Port S I/O, MISO of SPI0 |
| PS3 | TXD1 | — | — | — | $V_{DDX}$ | PERS/ PPSS | Up | Port S I/O, TXD of SCI1 |
| PS2 | RXD1 | — | — | — | $V_{DDX}$ | PERS/ PPSS | Up | Port S I/O, RXD of SCI1 |
| PS1 | TXD0 | — | — | — | $V_{DDX}$ | PERS/ PPSS | Up | Port S I/O, TXD of SCI0 |
| PS0 | RXD0 | — | — | — | $V_{DDX}$ | PERS/ PPSS | Up | Port S I/O, RXD of SCI0 |
| PT[7:6] | IOC[7:6] | — | — | — | $V_{DDX}$ | PERT/ PPST | Disabled | Port T I/O, ECT channels |
| PT[5] | IOC[5] | VREGAPI | — | — | $V_{DDX}$ | PERT/ PPST | Disabled | Port T I/O, ECT channels |
| PT[4:0] | IOC[4:0] | — | — | — | $V_{DDX}$ | PERT/ PPST | Disabled | Port T I/O, ECT channels |

## 1.2.3 Detailed Signal Descriptions

**NOTE**

The pin list of the largest package version of each MC9S12XE-Family derivative gives the complete of interface signals that also exist on smaller package options, although some of them are not bonded out. For devices assembled in smaller packages all non-bonded out pins should be configured as outputs after reset in order to avoid current drawn from floating inputs. Refer to Table 1-10 for affected pins. Particular attention is drawn to Port R, which does not have enabled pull-up/pull-down devices coming out of reset.

### 1.2.3.1 EXTAL, XTAL — Oscillator Pins

EXTAL and XTAL are the crystal driver and external clock pins. On reset all the device clocks are derived from the EXTAL input frequency. XTAL is the oscillator output.

### 1.2.3.2 $\overline{\text{RESET}}$ — External Reset Pin

The $\overline{\text{RESET}}$ pin is an active low bidirectional control signal. It acts as an input to initialize the MCU to a known start-up state. As an output it is driven low to indicate when any internal MCU reset source triggers. The $\overline{\text{RESET}}$ pin has an internal pull-up device.

### 1.2.3.3 TEST — Test Pin

This input only pin is reserved for test. This pin has a pull-down device.

**NOTE**

The TEST pin must be tied to $V_{SS}$ in all applications.

### 1.2.3.4 BKGD / MODC — Background Debug and Mode Pin

The BKGD/MODC pin is used as a pseudo-open-drain pin for the background debug communication. It is used as a MCU operating mode select pin during reset. The state of this pin is latched to the MODC bit at the rising edge of $\overline{\text{RESET}}$. The BKGD pin has a pull-up device.

### 1.2.3.5 PAD[15:0] / AN[15:0] — Port AD Input Pins of ATD0

PAD[15:0] are general-purpose input or output pins and analog inputs AN[15:0] of the analog-to-digital converter ATD0.

### 1.2.3.6 PAD[31:16] / AN[31:16] — Port AD Input Pins of ATD1

PAD[31:16] are general-purpose input or output pins and analog inputs AN[31:16] of the analog-to-digital converter ATD1.

### 1.2.3.7 PA[7:0] / ADDR[15:8] / IVD[15:8] — Port A I/O Pins

PA[7:0] are general-purpose input or output pins. In MCU expanded modes of operation, these pins are used for the external address bus. In MCU emulation modes of operation, these pins are used for external address bus and internal visibility read data.

### 1.2.3.8 PB[7:1] / ADDR[7:1] / IVD[7:1] — Port B I/O Pins

PB[7:1] are general-purpose input or output pins. In MCU expanded modes of operation, these pins are used for the external address bus. In MCU emulation modes of operation, these pins are used for external address bus and internal visibility read data.

### 1.2.3.9 PB0 / ADDR0 / $\overline{\text{UDS}}$ / IVD[0] — Port B I/O Pin 0

PB0 is a general-purpose input or output pin. In MCU expanded modes of operation, this pin is used for the external address bus ADDR0 or as upper data strobe signal. In MCU emulation modes of operation, this pin is used for external address bus ADDR0 and internal visibility read data IVD0.

### 1.2.3.10 PC[7:0] / DATA [15:8] — Port C I/O Pins

PC[7:0] are general-purpose input or output pins. In MCU expanded modes of operation, these pins are used for the external data bus.

The input voltage thresholds for PC[7:0] can be configured to reduced levels, to allow data from an external 3.3-V peripheral to be read by the MCU operating at 5.0 V. The input voltage thresholds for PC[7:0] are configured to reduced levels out of reset in expanded and emulation modes. The input voltage thresholds for PC[7:0] are configured to 5-V levels out of reset in normal modes.

### 1.2.3.11 PD[7:0] / DATA [7:0] — Port D I/O Pins

PD[7:0] are general-purpose input or output pins. In MCU expanded modes of operation, these pins are used for the external data bus.

The input voltage thresholds for PD[7:0] can be configured to reduced levels, to allow data from an external 3.3-V peripheral to be read by the MCU operating at 5.0 V. The input voltage thresholds for PD[7:0] are configured to reduced levels out of reset in expanded and emulation modes. The input voltage thresholds for PC[7:0] are configured to 5-V levels out of reset in normal modes.

### 1.2.3.12 PE7 / ECLKX2 / $\overline{\text{XCLKS}}$ — Port E I/O Pin 7

PE7 is a general-purpose input or output pin. ECLKX2 is a free running clock of twice the internal bus frequency, available by default in emulation modes and when enabled in other modes. The $\overline{\text{XCLKS}}$ is an input signal which controls whether a crystal in combination with the internal loop controlled Pierce oscillator is used or whether full swing Pierce oscillator/external clock circuitry is used (refer to Oscillator Configuration). An internal pullup is enabled during reset.

### 1.2.3.13 PE6 / MODB / $\overline{\text{TAGHI}}$ — Port E I/O Pin 6

PE6 is a general-purpose input or output pin. It is used as a MCU operating mode select pin during reset. The state of this pin is latched to the MODB bit at the rising edge of $\overline{\text{RESET}}$. This pin is an input with a pull-down device which is only active when $\overline{\text{RESET}}$ is low. $\overline{\text{TAGHI}}$ is used to tag the high half of the instruction word being read into the instruction queue.

The input voltage threshold for PE6 can be configured to reduced levels, to allow data from an external 3.3-V peripheral to be read by the MCU operating at 5.0 V. The input voltage threshold for PE6 is configured to reduced levels out of reset in expanded and emulation modes.

### 1.2.3.14 PE5 / MODA / $\overline{\text{TAGLO}}$ / $\overline{\text{RE}}$ — Port E I/O Pin 5

PE5 is a general-purpose input or output pin. It is used as an MCU operating mode select pin during reset. The state of this pin is latched to the MODA bit at the rising edge of $\overline{\text{RESET}}$. This pin is shared with the read enable $\overline{\text{RE}}$ output. This pin is an input with a pull-down device which is only active when $\overline{\text{RESET}}$ is low. $\overline{\text{TAGLO}}$ is used to tag the low half of the instruction word being read into the instruction queue.

The input voltage threshold for PE5 can be configured to reduced levels, to allow data from an external 3.3-V peripheral to be read by the MCU operating at 5.0 V. The input voltage threshold for PE5 is configured to reduced levels out of reset in expanded and emulation modes.

### 1.2.3.15 PE4 / ECLK — Port E I/O Pin 4

PE4 is a general-purpose input or output pin. It can be configured to drive the internal bus clock ECLK. ECLK can be used as a timing reference. The ECLK output has a programmable prescaler.

### 1.2.3.16 PE3 / $\overline{\text{LSTRB}}$ / $\overline{\text{LDS}}$ / EROMCTL— Port E I/O Pin 3

PE3 is a general-purpose input or output pin. In MCU expanded modes of operation, $\overline{\text{LSTRB}}$ or $\overline{\text{LDS}}$ can be used for the low byte strobe function to indicate the type of bus access. At the rising edge of $\overline{\text{RESET}}$ the state of this pin is latched to the EROMON bit.

### 1.2.3.17 PE2 / R/$\overline{\text{W}}$ / $\overline{\text{WE}}$— Port E I/O Pin 2

PE2 is a general-purpose input or output pin. In MCU expanded modes of operations, this pin drives the read/write output signal or write enable output signal for the external bus. It indicates the direction of data on the external bus.

### 1.2.3.18 PE1 / $\overline{\text{IRQ}}$ — Port E Input Pin 1

PE1 is a general-purpose input pin and the maskable interrupt request input that provides a means of applying asynchronous interrupt requests. This will wake up the MCU from stop or wait mode.

### 1.2.3.19 PE0 / $\overline{\text{XIRQ}}$ — Port E Input Pin 0

PE0 is a general-purpose input pin and the non-maskable interrupt request input that provides a means of applying asynchronous interrupt requests. This will wake up the MCU from stop or wait mode. The XIRQ

interrupt is level sensitive and active low. As XIRQ is level sensitive, while this pin is low the MCU will not enter STOP mode.

### 1.2.3.20 PF7 / TXD3 — Port F I/O Pin 7

PF7 is a general-purpose input or output pin. It can be configured as the transmit pin TXD of serial communication interface 3 (SCI3).

### 1.2.3.21 PF6 / RXD3 — Port F I/O Pin 6

PF6 is a general-purpose input or output pin. It can be configured as the transmit pin RXD of serial communication interface 3 (SCI3).

### 1.2.3.22 PF5 / SCL0 — Port F I/O Pin 5

PF5 is a general-purpose input or output pin. It can be configured as the serial clock pin SCL of the IIC0 module.

### 1.2.3.23 PF4 / SDA0 — Port F I/O Pin 4

PF4 is a general-purpose input or output pin. It can be configured as the serial data pin SDA of the IIC0 module.

### 1.2.3.24 PF[3:0] / $\overline{\text{CS[3:0]}}$ — Port F I/O Pins 3 to 0

PF[3:0] are a general-purpose input or output pins. They can be configured as chip select outputs [3:0].

### 1.2.3.25 PH7 / KWH7 / $\overline{\text{SS2}}$ / TXD5 — Port H I/O Pin 7

PH7 is a general-purpose input or output pin. It can be configured as a keypad wakeup input. It can be configured as slave select pin $\overline{\text{SS}}$ of the serial peripheral interface 2 (SPI2). It can be configured as the transmit pin TXD of serial communication interface 5 (SCI5).

### 1.2.3.26 PH6 / KWH6 / SCK2 / RXD5 — Port H I/O Pin 6

PH6 is a general-purpose input or output pin. It can be configured as a keypad wakeup input. It can be configured as serial clock pin SCK of the serial peripheral interface 2 (SPI2). It can be configured as the receive pin (RXD) of serial communication interface 5 (SCI5).

### 1.2.3.27 PH5 / KWH5 / MOSI2 / TXD4 — Port H I/O Pin 5

PH5 is a general-purpose input or output pin. It can be configured as a keypad wakeup input. It can be configured as master output (during master mode) or slave input pin (during slave mode) MOSI of the serial peripheral interface 2 (SPI2). It can be configured as the transmit pin TXD of serial communication interface 4 (SCI4).

### 1.2.3.28 PH4 / KWH4 / MISO2 / RXD4 — Port H I/O Pin 4

PH4 is a general-purpose input or output pin. It can be configured as a keypad wakeup input. It can be configured as master input (during master mode) or slave output (during slave mode) pin MISO of the serial peripheral interface 2 (SPI2). It can be configured as the receive pin RXD of serial communication interface 4 (SCI4).

### 1.2.3.29 PH3 / KWH3 / $\overline{SS1}$ — Port H I/O Pin 3

PH3 is a general-purpose input or output pin. It can be configured as a keypad wakeup input. It can be configured as slave select pin $\overline{SS}$ of the serial peripheral interface 1 (SPI1). It can also be configured as the transmit pin TXD of serial communication interface 7 (SCI7).

### 1.2.3.30 PH2 / KWH2 / SCK1 — Port H I/O Pin 2

PH2 is a general-purpose input or output pin. It can be configured as a keypad wakeup input. It can be configured as serial clock pin SCK of the serial peripheral interface 1 (SPI1). It can be configured as the receive pin RXD of serial communication interface 7 (SCI7).

### 1.2.3.31 PH1 / KWH1 / MOSI1 — Port H I/O Pin 1

PH1 is a general-purpose input or output pin. It can be configured as a keypad wakeup input. It can be configured as master output (during master mode) or slave input pin (during slave mode) MOSI of the serial peripheral interface 1 (SPI1). It can also be configured as the transmit pin TXD of serial communication interface 6 (SCI6).

### 1.2.3.32 PH0 / KWH0 / MISO1 — Port H I/O Pin 0

PH0 is a general-purpose input or output pin. It can be configured as a keypad wakeup input. It can be configured as master input (during master mode) or slave output (during slave mode) pin MISO of the serial peripheral interface 1 (SPI1). It can be configured as the receive pin RXD of serial communication interface 6 (SCI6).

### 1.2.3.33 PJ7 / KWJ7 / TXCAN4 / SCL0 / TXCAN0— PORT J I/O Pin 7

PJ7 is a general-purpose input or output pin. It can be configured as a keypad wakeup input. It can be configured as the transmit pin TXCAN for the scalable controller area network controller 0 or 4 (CAN0 or CAN4) or as the serial clock pin SCL of the IIC0 module.

### 1.2.3.34 PJ6 / KWJ6 / RXCAN4 / SDA0 / RXCAN0 — PORT J I/O Pin 6

PJ6 is a general-purpose input or output pin. It can be configured as a keypad wakeup input. It can be configured as the receive pin RXCAN for the scalable controller area network controller 0 or 4 (CAN0 or CAN4) or as the serial data pin SDA of the IIC0 module.

### 1.2.3.35 PJ5 / KWJ5 / SCL1 / $\overline{CS2}$ — PORT J I/O Pin 5

PJ5 is a general-purpose input or output pin. It can be configured as a keypad wakeup input. It can be configured as the serial clock pin SCL of the IIC1 module. It can be also configured as chip-select output 2.

### 1.2.3.36 PJ4 / KWJ4 / SDA1 / $\overline{CS0}$ — PORT J I/O Pin 4

PJ4 is a general-purpose input or output pin. It can be configured as a keypad wakeup input. It can be configured as the serial data pin SDA of the IIC1 module. It can also be configured as chip-select output.

### 1.2.3.37 PJ3 / KWJ3 — PORT J I/O Pin 3

PJ3 is a general-purpose input or output pins. It can be configured as a keypad wakeup input.

### 1.2.3.38 PJ2 / KWJ2 / $\overline{CS1}$ — PORT J I/O Pin 2

PJ2 is a general-purpose input or output pins. It can be configured as a keypad wakeup input. It can also be configured as chip-select output.

### 1.2.3.39 PJ1 / KWJ1 / TXD2 — PORT J I/O Pin 1

PJ1 is a general-purpose input or output pin. It can be configured as a keypad wakeup input. It can be configured as the transmit pin TXD of the serial communication interface 2 (SCI2).

### 1.2.3.40 PJ0 / KWJ0 / RXD2 / $\overline{CS3}$ — PORT J I/O Pin 0

PJ0 is a general-purpose input or output pin. It can be configured as a keypad wakeup input. It can be configured as the receive pin RXD of the serial communication interface 2 (SCI2).It can also be configured as chip-select output 3.

### 1.2.3.41 PK7 / $\overline{EWAIT}$ / ROMCTL — Port K I/O Pin 7

PK7 is a general-purpose input or output pin. During MCU emulation modes and normal expanded modes of operation, this pin is used to enable the Flash EEPROM memory in the memory map (ROMCTL). At the rising edge of $\overline{RESET}$, the state of this pin is latched to the ROMON bit. The $\overline{EWAIT}$ input signal maintains the external bus access until the external device is ready to capture data (write) or provide data (read).

The input voltage threshold for PK7 can be configured to reduced levels, to allow data from an external 3.3-V peripheral to be read by the MCU operating at 5.0 V.

### 1.2.3.42 PK[6:4] / ADDR[22:20] / ACC[2:0] — Port K I/O Pin [6:4]

PK[6:4] are general-purpose input or output pins. During MCU expanded modes of operation, the ACC[2:0] signals are used to indicate the access source of the bus cycle. These pins also provide the expanded addresses ADDR[22:20] for the external bus. In Emulation modes ACC[2:0] is available and is time multiplexed with the high addresses

### 1.2.3.43 PK[3:0] / ADDR[19:16] / IQSTAT[3:0] — Port K I/O Pins [3:0]

PK3-PK0 are general-purpose input or output pins. In MCU expanded modes of operation, these pins provide the expanded address ADDR[19:16] for the external bus and carry instruction pipe information.

### 1.2.3.44 PL7 / TXD7 — Port L I/O Pin 7

PL7 is a general-purpose input or output pin. It can be configured as the transmit pin TXD of serial communication interface 7 (SCI7).

### 1.2.3.45 PL6 / RXD7 — Port L I/O Pin 6

PL6 is a general-purpose input or output pin. It can be configured as the receive pin RXD of serial communication interface 7 (SCI7).

### 1.2.3.46 PL5 / TXD6 — Port L I/O Pin 5

PL5 is a general-purpose input or output pin. It can be configured as the transmit pin TXD of serial communication interface 6 (SCI6).

### 1.2.3.47 PL4 / RXD6 — Port L I/O Pin 4

PL4 is a general-purpose input or output pin. It can be configured as the receive pin RXD of serial communication interface 6 (SCI6).

### 1.2.3.48 PL3 / TXD5 — Port L I/O Pin 3

PL3 is a general-purpose input or output pin. It can be configured as the transmit pin TXD of serial communication interface 5 (SCI5).

### 1.2.3.49 PL2 / RXD5 — Port L I/O Pin 2

PL2 is a general-purpose input or output pin. It can be configured as the receive pin RXD of serial communication interface 5 (SCI5).

### 1.2.3.50 PL1 / TXD4 — Port L I/O Pin 1

PL1 is a general-purpose input or output pin. It can be configured as the transmit pin TXD of serial communication interface 4 (SCI4).

### 1.2.3.51 PL0 / RXD4 — Port L I/O Pin 0

PL0 is a general-purpose input or output pin. It can be configured as the receive pin RXD of serial communication interface 4 (SCI4).

### 1.2.3.52 PM7 / TXCAN3 / TXCAN4 / TXD3 — Port M I/O Pin 7

PM7 is a general-purpose input or output pin. It can be configured as the transmit pin TXCAN of the scalable controller area network controller 3 or 4 (CAN3 or CAN4). PM7 can be configured as the transmit pin TXD3 of the serial communication interface 3 (SCI3).

### 1.2.3.53 PM6 / RXCAN3 / RXCAN4 / RXD3 — Port M I/O Pin 6

PM6 is a general-purpose input or output pin. It can be configured as the receive pin RXCAN of the scalable controller area network controller 3 or 4 (CAN3 or CAN4). PM6 can be configured as the receive pin RXD3 of the serial communication interface 3 (SCI3).

### 1.2.3.54 PM5 / TXCAN0 / TXCAN2 / TXCAN4 / SCK0 — Port M I/O Pin 5

PM5 is a general-purpose input or output pin. It can be configured as the transmit pin TXCAN of the scalable controller area network controllers 0, 2 or 4 (CAN0, CAN2, or CAN4). It can be configured as the serial clock pin SCK of the serial peripheral interface 0 (SPI0).

### 1.2.3.55 PM4 / RXCAN0 / RXCAN2 / RXCAN4 / MOSI0 — Port M I/O Pin 4

PM4 is a general-purpose input or output pin. It can be configured as the receive pin RXCAN of the scalable controller area network controllers 0, 2, or 4 (CAN0, CAN2, or CAN4). It can be configured as the master output (during master mode) or slave input pin (during slave mode) MOSI for the serial peripheral interface 0 (SPI0).

### 1.2.3.56 PM3 / TXCAN1 / TXCAN0 / $\overline{SS0}$ — Port M I/O Pin 3

PM3 is a general-purpose input or output pin. It can be configured as the transmit pin TXCAN of the scalable controller area network controllers 1 or 0 (CAN1 or CAN0). It can be configured as the slave select pin $\overline{SS}$ of the serial peripheral interface 0 (SPI0).

### 1.2.3.57 PM2 / RXCAN1 / RXCAN0 / MISO0 — Port M I/O Pin 2

PM2 is a general-purpose input or output pin. It can be configured as the receive pin RXCAN of the scalable controller area network controllers 1 or 0 (CAN1 or CAN0). It can be configured as the master input (during master mode) or slave output pin (during slave mode) MISO for the serial peripheral interface 0 (SPI0).

### 1.2.3.58 PM1 / TXCAN0 — Port M I/O Pin 1

PM1 is a general-purpose input or output pin. It can be configured as the transmit pin TXCAN of the scalable controller area network controller 0 (CAN0).

### 1.2.3.59 PM0 / RXCAN0 — Port M I/O Pin 0

PM0 is a general-purpose input or output pin. It can be configured as the receive pin RXCAN of the scalable controller area network controller 0 (CAN0).

### 1.2.3.60 PP7 / KWP7 / PWM7 / SCK2 / TIMIOC7— Port P I/O Pin 7

PP7 is a general-purpose input or output pin. It can be configured as a keypad wakeup input. It can be configured as pulse width modulator (PWM) channel 7 output, TIM channel 7, or as serial clock pin SCK of the serial peripheral interface 2 (SPI2).

### 1.2.3.61 PP6 / KWP6 / PWM6 / $\overline{SS2}$ / TIMIOC6— Port P I/O Pin 6

PP6 is a general-purpose input or output pin. It can be configured as a keypad wakeup input. It can be configured as pulse width modulator (PWM) channel 6 output, TIM channel 6 or as the slave select pin $\overline{SS}$ of the serial peripheral interface 2 (SPI2).

### 1.2.3.62 PP5 / KWP5 / PWM5 / MOSI2 / TIMIOC5— Port P I/O Pin 5

PP5 is a general-purpose input or output pin. It can be configured as a keypad wakeup input. It can be configured as pulse width modulator (PWM) channel 5 output, TIM channel 5 or as the master output (during master mode) or slave input pin (during slave mode) MOSI of the serial peripheral interface 2 (SPI2).

### 1.2.3.63 PP4 / KWP4 / PWM4 / MISO2 / TIMIOC4— Port P I/O Pin 4

PP4 is a general-purpose input or output pin. It can be configured as a keypad wakeup input. It can be configured as pulse width modulator (PWM) channel 4 output, TIM channel 4 or as the master input (during master mode) or slave output (during slave mode) pin MISO of the serial peripheral interface 2 (SPI2).

### 1.2.3.64 PP3 / KWP3 / PWM3 / $\overline{SS1}$ / TIMIOC3— Port P I/O Pin 3

PP3 is a general-purpose input or output pin. It can be configured as a keypad wakeup input. It can be configured as pulse width modulator (PWM) channel 3 output, TIM channel 3, or as the slave select pin $\overline{SS}$ of the serial peripheral interface 1 (SPI1).

### 1.2.3.65 PP2 / KWP2 / PWM2 / SCK1 / TIMIOC2— Port P I/O Pin 2

PP2 is a general-purpose input or output pin. It can be configured as a keypad wakeup input. It can be configured as pulse width modulator (PWM) channel 2 output, TIM channel 2, or as the serial clock pin SCK of the serial peripheral interface 1 (SPI1).

### 1.2.3.66 PP1 / KWP1 / PWM1 / MOSI1 / TIMIOC1— Port P I/O Pin 1

PP1 is a general-purpose input or output pin. It can be configured as a keypad wakeup input. It can be configured as pulse width modulator (PWM) channel 1 output, TIM channel 1, or master output (during master mode) or slave input pin (during slave mode) MOSI of the serial peripheral interface 1 (SPI1).

### 1.2.3.67 PP0 / KWP0 / PWM0 / MISO1 / TIMIOC0— Port P I/O Pin 0

PP0 is a general-purpose input or output pin. It can be configured as a keypad wakeup input. It can be configured as pulse width modulator (PWM) channel 0 output, TIM channel 0 or as the master input (during master mode) or slave output (during slave mode) pin MISO of the serial peripheral interface 1 (SPI1).

### 1.2.3.68 PR[7:0] / TIMIOC[7:0] — Port R I/O Pins [7:0]

PR[7:0] are general-purpose input or output pins. They can be configured as input capture or output compare pins IOC[7:0] of the standard timer (TIM).

### 1.2.3.69 PS7 / $\overline{SS0}$ — Port S I/O Pin 7

PS7 is a general-purpose input or output pin. It can be configured as the slave select pin $\overline{SS}$ of the serial peripheral interface 0 (SPI0).

### 1.2.3.70 PS6 / SCK0 — Port S I/O Pin 6

PS6 is a general-purpose input or output pin. It can be configured as the serial clock pin SCK of the serial peripheral interface 0 (SPI0).

### 1.2.3.71 PS5 / MOSI0 — Port S I/O Pin 5

PS5 is a general-purpose input or output pin. It can be configured as master output (during master mode) or slave input pin (during slave mode) MOSI of the serial peripheral interface 0 (SPI0).

### 1.2.3.72 PS4 / MISO0 — Port S I/O Pin 4

PS4 is a general-purpose input or output pin. It can be configured as master input (during master mode) or slave output pin (during slave mode) MOSI of the serial peripheral interface 0 (SPI0).

### 1.2.3.73 PS3 / TXD1 — Port S I/O Pin 3

PS3 is a general-purpose input or output pin. It can be configured as the transmit pin TXD of serial communication interface 1 (SCI1).

### 1.2.3.74 PS2 / RXD1 — Port S I/O Pin 2

PS2 is a general-purpose input or output pin. It can be configured as the receive pin RXD of serial communication interface 1 (SCI1).

### 1.2.3.75 PS1 / TXD0 — Port S I/O Pin 1

PS1 is a general-purpose input or output pin. It can be configured as the transmit pin TXD of serial communication interface 0 (SCI0).

#### 1.2.3.76 PS0 / RXD0 — Port S I/O Pin 0

PS0 is a general-purpose input or output pin. It can be configured as the receive pin RXD of serial communication interface 0 (SCI0).

#### 1.2.3.77 PT[7:6] / IOC[7:6] — Port T I/O Pins [7:6]

PT[7:6] are general-purpose input or output pins. They can be configured as input capture or output compare pins IOC[7:6] of the enhanced capture timer (ECT).

#### 1.2.3.78 PT[5] / IOC[5] / VREG_API— Port T I/O Pins [5]

PT[5] is a general-purpose input or output pin. It can be configured as input capture or output compare pin IOC[5] of the enhanced capture timer (ECT) or can be configured to output the VREG_API signal.

#### 1.2.3.79 PT[4:0] / IOC[4:0] — Port T I/O Pins [4:0]

PT[4:0] are general-purpose input or output pins. They can be configured as input capture or output compare pins IOC[4:0] of the enhanced capture timer (ECT).

### 1.2.4 Power Supply Pins

MC9S12XE-Family power and ground pins are described below.

Because fast signal transitions place high, short-duration current demands on the power supply, use bypass capacitors with high-frequency characteristics and place them as close to the MCU as possible.

**NOTE**

All $V_{SS}$ pins must be connected together in the application.

#### 1.2.4.1 VDDX[7:1], VSSX[7:1] — Power and Ground Pins for I/O Drivers

External power and ground for I/O drivers. Bypass requirements depend on how heavily the MCU pins are loaded. All $V_{DDX}$ pins are connected together internally. All $V_{SSX}$ pins are connected together internally.

#### 1.2.4.2 VDDR — Power Pin for Internal Voltage Regulator

Input to the internal voltage regulator. The internal voltage regulator is turned off, if $V_{DDR}$ is tied to ground

#### 1.2.4.3 VDD, VSS1,VSS2,VSS3 — Core Power Pins

Power is supplied to the MCU core from the internal voltage regulator, whose load capacitor must be connected to VDD. The voltage supply of nominally 1.8V is derived from the internal voltage regulator. The return current path is through the VSS1,VSS2 and VSS3 pins. No static external loading of these pins is permitted.

### 1.2.4.4 VDDF — NVM Power Pin

Power is supplied to the MCU NVM through VDDF . The voltage supply of nominally 2.8V is derived from the internal voltage regulator. No static external loading of these pins is permitted.

### 1.2.4.5 VDDA2, VDDA1, VSSA2, VSSA1 — Power Supply Pins for ATD and Voltage Regulator

These are the power supply and ground input pins for the analog-to-digital converters and the voltage regulator. Internally the $V_{DDA}$ pins are connected together. Internally the $V_{SSA}$ pins are connected together.

### 1.2.4.6 VRH, VRL — ATD Reference Voltage Input Pins

$V_{RH}$ and $V_{RL}$ are the reference voltage input pins for the analog-to-digital converter.

### 1.2.4.7 VDDPLL, VSSPL — Power Supply Pins for PLL

These pins provide operating voltage and ground for the oscillator and the phased-locked loop. The voltage supply of nominally 1.8V is derived from the internal voltage regulator. This allows the supply voltage to the oscillator and PLL to be bypassed independently. This voltage is generated by the internal voltage regulator. No static external loading of these pins is permitted.

**Table 1-11. Power and Ground Connection Summary**

| Mnemonic | Nominal Voltage | Description |
|---|---|---|
| VDDR | 5.0 V | External power supply to internal voltage regulator |
| VDDX[7:1] | 5.0 V | External power and ground, supply to pin drivers |
| VSSX[7:1] | 0 V | |
| VDDA2, VDDA1 | 5.0 V | Operating voltage and ground for the analog-to-digital converters and the reference for the internal voltage regulator, allows the supply voltage to the A/D to be bypassed independently. |
| VSSA2, VSSA1 | 0 V | |
| VRL | 0 V | Reference voltages for the analog-to-digital converter. |
| VRH | 5.0 V | |
| VDD | 1.8 V | Internal power and ground generated by internal regulator for the internal core. |
| VSS1, VSS2, VSS3 | 0V | |
| VDDF | 2.8 V | Internal power and ground generated by internal regulator for the internal NVM. |

**Table 1-11. Power and Ground Connection Summary (continued)**

| Mnemonic | Nominal Voltage | Description |
|---|---|---|
| VDDPLL | 1.8 V | Provides operating voltage and ground for the phased-locked loop. This allows the supply voltage to the PLL to be bypassed independently. Internal power and ground generated by internal regulator. |
| VSSPLL | 0 V | |

## 1.3 System Clock Description

The clock and reset generator module (CRG) provides the internal clock signals for the core and all peripheral modules. Figure 1-9 shows the clock connections from the CRG to all modules.

Consult the CRG specification for details on clock generation.

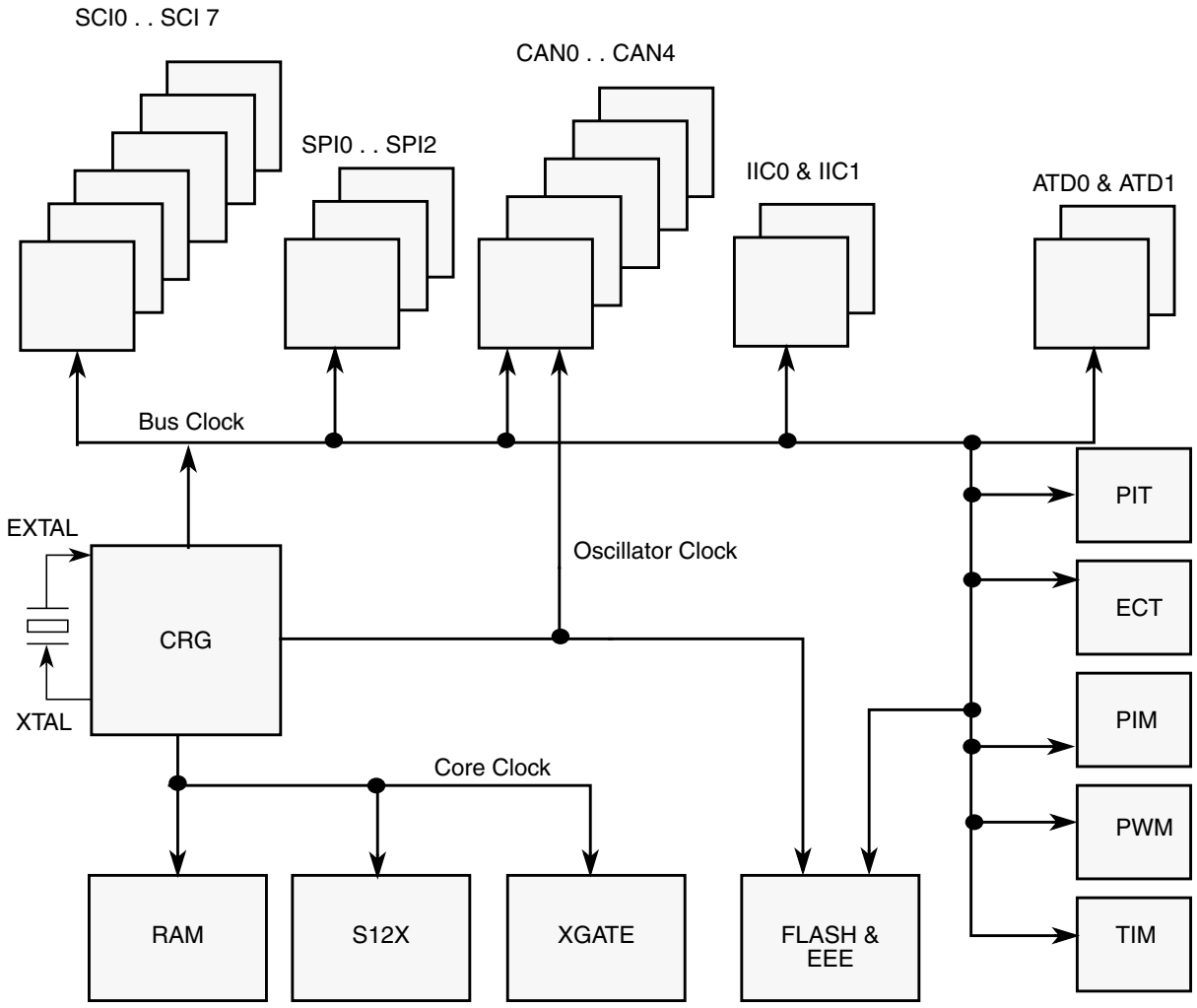



**Figure 1-9. Clock Connections**

The system clock can be supplied in several ways enabling a range of system operating frequencies to be supported:

- The on-chip phase locked loop (PLL)
- the PLL self clocking
- the oscillator

The clock generated by the PLL or oscillator provides the main system clock frequencies core clock and bus clock. As shown in Figure 1-9, these system clocks are used throughout the MCU to drive the core, the memories, and the peripherals.

The program Flash memory and the EEPROM are supplied by the bus clock and the oscillator clock. The oscillator clock is used as a time base to derive the program and erase times for the NVM's.

The CAN modules may be configured to have their clock sources derived either from the bus clock or directly from the oscillator clock. This allows the user to select its clock based on the required jitter performance.

In order to ensure the presence of the clock the MCU includes an on-chip clock monitor connected to the output of the oscillator. The clock monitor can be configured to invoke the PLL self-clocking mode or to generate a system reset if it is allowed to time out as a result of no oscillator clock being present.

In addition to the clock monitor, the MCU also provides a clock quality checker which performs a more accurate check of the clock. The clock quality checker counts a predetermined number of clock edges within a defined time window to insure that the clock is running. The checker can be invoked following specific events such as on wake-up or clock monitor failure.

## 1.4 Modes of Operation

The MCU can operate in different modes associated with MCU resource mapping and bus interface configuration. These are described in 1.4.1 Chip Configuration Summary.

The MCU can operate in different power modes to facilitate power saving when full system performance is not required. These are described in 1.4.2 Power Modes.

Some modules feature a software programmable option to freeze the module status whilst the background debug module is active to facilitate debugging. This is described in 1.4.3 Freeze Mode.

For system integrity support separate system states are featured as explained in 1.4.4 System States.

### 1.4.1 Chip Configuration Summary

The MCU can operate in six different modes associated with resource configuration. The different modes, the state of ROMCTL and EROMCTL signal on rising edge of $\overline{\text{RESET}}$ and the security state of the MCU affect the following device characteristics:

- External bus interface configuration
- Flash in memory map, or not
- Debug features enabled or disabled

The operating mode out of reset is determined by the states of the MODC, MODB, and MODA signals during reset (see Table 1-12). The MODC, MODB, and MODA bits in the MODE register show the current operating mode and provide limited mode switching during operation. The states of the MODC, MODB, and MODA signals are latched into these bits on the rising edge of $\overline{\text{RESET}}$.

In normal expanded mode and in emulation modes the ROMON bit and the EROMON bit in the MMCCTL1 register defines if the on chip flash memory is the memory map, or not. (See Table 1-12.) For a detailed explanation of the ROMON and EROMON bits refer to the MMC description.

The state of the ROMCTL signal is latched into the ROMON bit in the MMCCTL1 register on the rising edge of $\overline{\text{RESET}}$. The state of the EROMCTL signal is latched into the EROMON bit in the MMCCTL1 register on the rising edge of $\overline{\text{RESET}}$.

**Table 1-12. Chip Modes and Data Sources**

| Chip Modes | MODC | MODB | MODA | ROMCTL | EROMCTL | Data Source[1] |
|---|---|---|---|---|---|---|
| Normal single chip | 1 | 0 | 0 | X | X | Internal |
| Special single chip | 0 | 0 | 0 | | | |
| Emulation single chip | 0 | 0 | 1 | X | 0 | Emulation memory |
| | | | | X | 1 | Internal Flash |
| Normal expanded | 1 | 0 | 1 | 0 | X | External application |
| | | | | 1 | X | Internal Flash |
| Emulation expanded | 0 | 1 | 1 | 0 | X | External application |
| | | | | 1 | 0 | Emulation memory |
| | | | | 1 | 1 | Internal Flash |
| Special test | 0 | 1 | 0 | 0 | X | External application |
| | | | | 1 | X | Internal Flash |

1. Internal means resources inside the MCU are read/written.
Internal Flash means Flash resources inside the MCU are read/written.
Emulation memory means resources inside the emulator are read/written (PRU registers, Flash replacement, RAM, EEPROM, and register space are always considered internal).
External application means resources residing outside the MCU are read/written.

#### 1.4.1.1 Normal Expanded Mode

Ports K, A, and B are configured as a 23-bit address bus, ports C and D are configured as a 16-bit data bus, and port E provides bus control and status signals. This mode allows 16-bit external memory and peripheral devices to be interfaced to the system. The fastest external bus rate is divide by 2 from the internal bus rate.

#### 1.4.1.2 Normal Single-Chip Mode

There is no external bus in this mode. The processor program is executed from internal memory. Ports A, B,C,D, K, and most pins of port E are available as general-purpose I/O.

#### 1.4.1.3 Special Single-Chip Mode

This mode is used for debugging single-chip operation, boot-strapping, or security related operations. The background debug module BDM is active in this mode. The CPU executes a monitor program located in an on-chip ROM. BDM firmware waits for additional serial commands through the BKGD pin. There is no external bus after reset in this mode.

#### 1.4.1.4 Emulation of Expanded Mode

Developers use this mode for emulation systems in which the users target application is normal expanded mode. Code is executed from external memory or from internal memory depending on the state of ROMON and EROMON bit. In this mode the internal operation is visible on external bus interface.

#### 1.4.1.5 Emulation of Single-Chip Mode

Developers use this mode for emulation systems in which the user's target application is normal single-chip mode. Code is executed from external memory or from internal memory depending on the state of ROMON and EROMON bit. In this mode the internal operation is visible on external bus interface.

#### 1.4.1.6 Special Test Mode

This is for Freescale internal use only.

### 1.4.2 Power Modes

The MCU features two main low-power modes. Consult the respective module description for module specific behavior in system stop, system pseudo stop, and system wait mode. An important source of information about the clock system is the Clock and Reset Generator description (CRG).

#### 1.4.2.1 System Stop Modes

The system stop modes are entered if the CPU executes the STOP instruction unless either the XGATE is active or an NVM command is active. The XGATE is active if it executes a thread or the XGFACT bit in the XGMCTL register is set. Depending on the state of the PSTP bit in the CLKSEL register the MCU goes into pseudo stop mode or full stop mode. Please refer to CRG description. Asserting $\overline{\text{RESET}}$, $\overline{\text{XIRQ}}$, $\overline{\text{IRQ}}$ or any other interrupt that is not masked exits system stop modes. System stop modes can be exited by XGATE or CPU activity independently, depending on the configuration of the interrupt request. If System-Stop is exited on an XGATE request then, as long as the XGATE does not set an interrupt flag on the CPU and the XGATE fake activity bit (FACT) remains cleared, once XGATE activity is completed System Stop mode will automatically be re-entered.

If the CPU executes the STOP instruction whilst XGATE is active or an NVM command is being processed, then the system clocks continue running until XGATE/NVM activity is completed. If a non-masked interrupt occurs within this time then the system does not effectively enter stop mode although the STOP instruction has been executed.

#### 1.4.2.2 Full Stop Mode

The oscillator is stopped in this mode. By default all clocks are switched off and all counters and dividers remain frozen. The Autonomous Periodic Interrupt (API) and ATD modules may be enabled to self wake the device. A Fast wake up mode is available to allow the device to wake from Full Stop mode immediately on the PLL internal clock without starting the oscillator clock.

#### 1.4.2.3 Pseudo Stop Mode

In this mode the system clocks are stopped but the oscillator is still running and the real time interrupt (RTI) and watchdog (COP), API and ATD modules may be enabled. Other peripherals are turned off. This mode consumes more current than system stop mode but, as the oscillator continues to run, the full speed wake up time from this mode is significantly shorter.

#### 1.4.2.4 XGATE Fake Activity Mode

This mode is entered if the CPU executes the STOP instruction when the XGATE is not executing a thread and the XGFACT bit in the XGMCTL register is set. The oscillator remains active and any enabled peripherals continue to function.

#### 1.4.2.5 Wait Mode

This mode is entered when the CPU executes the WAI instruction. In this mode the CPU will not execute instructions. The internal CPU clock is switched off. All peripherals and the XGATE can be active in system wait mode. For further power consumption the peripherals can individually turn off their local clocks. Asserting $\overline{\text{RESET}}$, $\overline{\text{XIRQ}}$, $\overline{\text{IRQ}}$ or any other interrupt that is not masked and is not routed to XGATE ends system wait mode.

#### 1.4.2.6 Run Mode

Although this is not a low-power mode, unused peripheral modules should not be enabled in order to save power.

### 1.4.3 Freeze Mode

The enhanced capture timer, pulse width modulator, analog-to-digital converters, and the periodic interrupt timer provide a software programmable option to freeze the module status when the background debug module is active. This is useful when debugging application software. For detailed description of the behavior of the ATD0, ATD1, ECT, PWM, and PIT when the background debug module is active consult the corresponding Block Guides.

### 1.4.4 System States

To facilitate system integrity the MCU can run in Supervisor state or User state. The System States strategy is implemented by additional features on the S12X CPU and a Memory Protection Unit. This is designed to support restricted access for code modules executed by kernels or operating systems supporting access control to system resources.

The current system state is indicated by the U bit in the CPU condition code register. In User state certain CPU instructions are restricted. See the CPU reference guide for details of the U bit and of those instructions affected by User state.

In the case that software task accesses resources outside those defined for it in the MPU a non-maskable interrupt is generated.

#### 1.4.4.1 Supervisor State

This state is intended for configuring the MPU for different tasks that are then executed in User state, returning to Supervisor state on completion of each task. This is the default ’state’ following reset and can be re-entered from User state by an exception (interrupt). If the SVSEN bit in the MPUSEL register of the

MPU is set, access to system resources is only allowed if enabled by a memory range descriptor as defined in the Memory Protection Unit (MPU) description.

#### 1.4.4.2 User State

This state is intended for carrying out system tasks and is entered by setting the U bit of the condition codes register while in Supervisor state. Restrictions apply for the execution of several CPU instructions in User state and access to system resources is only allowed in if enabled by a memory range descriptor as defined in the Memory Protection Unit (MPU) description.

## 1.5 Security

The MCU security feature allows the protection of the on chip Flash and emulated EEPROM memory. For a detailed description of the security features refer to the S12X9SEC description.

## 1.6 Resets and Interrupts

Consult the S12XCPU manual and the S12XINT description for information on exception processing.

### 1.6.1 Resets

Resets are explained in detail in the Clock Reset Generator (CRG) description.

**Table 1-13. Reset Sources and Vector Locations**

| Vector Address | Reset Source | CCR Mask | Local Enable |
|---|---|---|---|
| $FFFE | Power-On Reset (POR) | None | None |
| $FFFE | Low Voltage Reset (LVR) | None | None |
| $FFFE | External pin $\overline{\text{RESET}}$ | None | None |
| $FFFE | Illegal Address Reset | None | None |
| $FFFC | Clock monitor reset | None | PLLCTL (CME, SCME) |
| $FFFA | COP watchdog reset | None | COP rate select |

### 1.6.2 Vectors

Table 1-14 lists all interrupt sources and vectors in the default order of priority. The interrupt module (S12XINT) provides an interrupt vector base register (IVBR) to relocate the vectors. Associated with each I-bit maskable service request is a configuration register. It selects if the service request is enabled, the service request priority level and whether the service request is handled either by the S12X CPU or by the XGATE module.

**Table 1-14. Interrupt Vector Locations (Sheet 1 of 4)**

| Vector Address[1] | XGATE Channel ID[2] | Interrupt Source | CCR Mask | Local Enable | STOP Wake up | WAIT Wake up |
|---|---|---|---|---|---|---|
| Vector base + $F8 | — | Unimplemented instruction trap | None | None | — | — |
| Vector base+ $F6 | — | SWI | None | None | — | — |
| Vector base+ $F4 | — | $\overline{\text{XIRQ}}$ | X Bit | None | Yes | Yes |
| Vector base+ $F2 | — | $\overline{\text{IRQ}}$ | I bit | IRQCR (IRQEN) | Yes | Yes |
| Vector base+ $F0 | $78 | Real time interrupt | I bit | CRGINT (RTIE) | Refer to CRG interrupt section | |
| Vector base+ $EE | $77 | Enhanced capture timer channel 0 | I bit | TIE (C0I) | No | Yes |
| Vector base + $EC | $76 | Enhanced capture timer channel 1 | I bit | TIE (C1I) | No | Yes |
| Vector base+ $EA | $75 | Enhanced capture timer channel 2 | I bit | TIE (C2I) | No | Yes |
| Vector base+ $E8 | $74 | Enhanced capture timer channel 3 | I bit | TIE (C3I) | No | Yes |
| Vector base+ $E6 | $73 | Enhanced capture timer channel 4 | I bit | TIE (C4I) | No | Yes |
| Vector base+ $E4 | $72 | Enhanced capture timer channel 5 | I bit | TIE (C5I) | No | Yes |
| Vector base + $E2 | $71 | Enhanced capture timer channel 6 | I bit | TIE (C6I) | No | Yes |
| Vector base+ $E0 | $70 | Enhanced capture timer channel 7 | I bit | TIE (C7I) | No | Yes |
| Vector base+ $DE | $6F | Enhanced capture timer overflow | I bit | TSRC2 (TOF) | No | Yes |
| Vector base+ $DC | $6E | Pulse accumulator A overflow | I bit | PACTL (PAOVI) | No | Yes |
| Vector base + $DA | $6D | Pulse accumulator input edge | I bit | PACTL (PAI) | No | Yes |
| Vector base + $D8 | $6C | SPI0 | I bit | SPI0CR1 (SPIE, SPTIE) | No | Yes |
| Vector base+ $D6 | $6B | SCI0 | I bit | SCI0CR2 (TIE, TCIE, RIE, ILIE) | Yes | Yes |
| Vector base + $D4 | $6A | SCI1 | I bit | SCI1CR2 (TIE, TCIE, RIE, ILIE) | Yes | Yes |
| Vector base + $D2 | $69 | ATD0 | I bit | ATD0CTL2 (ASCIE) | Yes | Yes |
| Vector base + $D0 | $68 | ATD1 | I bit | ATD1CTL2 (ASCIE) | Yes | Yes |
| Vector base + $CE | $67 | Port J | I bit | PIEJ (PIEJ7-PIEJ0) | Yes | Yes |
| Vector base + $CC | $66 | Port H | I bit | PIEH (PIEH7-PIEH0) | Yes | Yes |
| Vector base + $CA | $65 | Modulus down counter underflow | I bit | MCCTL(MCZI) | No | Yes |
| Vector base + $C8 | $64 | Pulse accumulator B overflow | I bit | PBCTL(PBOVI) | No | Yes |
| Vector base + $C6 | $63 | CRG PLL lock | I bit | CRGINT(LOCKIE) | Refer to CRG interrupt section | |
| Vector base + $C4 | $62 | CRG self-clock mode | I bit | CRGINT (SCMIE) | Refer to CRG interrupt section | |
| Vector base + $C2 | $61 | SCI6 | I bit | SCI6CR2 (TIE, TCIE, RIE, ILIE) | Yes | Yes |
| Vector base + $C0 | $60 | IIC0 bus | I bit | IBCR0 (IBIE) | No | Yes |

**Table 1-14. Interrupt Vector Locations (Sheet 2 of 4)**

| Vector Address[1] | XGATE Channel ID[2] | Interrupt Source | CCR Mask | Local Enable | STOP Wake up | WAIT Wake up |
|---|---|---|---|---|---|---|
| Vector base + $BE | $5F | SPI1 | I bit | SPI1CR1 (SPIE, SPTIE) | No | Yes |
| Vector base + $BC | $5E | SPI2 | I bit | SPI2CR1 (SPIE, SPTIE) | No | Yes |
| Vector base + $BA | $5D | FLASH Fault Detect | I bit | FCNFG2 (FDIE) | No | No |
| Vector base + $B8 | $5C | FLASH | I bit | FCNFG (CCIE, CBEIE) | No | Yes |
| Vector base + $B6 | $5B | CAN0 wake-up | I bit | CAN0RIER (WUPIE) | Yes | Yes |
| Vector base + $B4 | $5A | CAN0 errors | I bit | CAN0RIER (CSCIE, OVRIE) | No | Yes |
| Vector base + $B2 | $59 | CAN0 receive | I bit | CAN0RIER (RXFIE) | No | Yes |
| Vector base + $B0 | $58 | CAN0 transmit | I bit | CAN0TIER (TXEIE[2:0]) | No | Yes |
| Vector base + $AE | $57 | CAN1 wake-up | I bit | CAN1RIER (WUPIE) | Yes | Yes |
| Vector base + $AC | $56 | CAN1 errors | I bit | CAN1RIER (CSCIE, OVRIE) | No | Yes |
| Vector base + $AA | $55 | CAN1 receive | I bit | CAN1RIER (RXFIE) | No | Yes |
| Vector base + $A8 | $54 | CAN1 transmit | I bit | CAN1TIER (TXEIE[2:0]) | No | Yes |
| Vector base + $A6 | $53 | CAN2 wake-up | I bit | CAN2RIER (WUPIE) | Yes | Yes |
| Vector base + $A4 | $52 | CAN2 errors | I bit | CAN2RIER (CSCIE, OVRIE) | No | Yes |
| Vector base + $A2 | $51 | CAN2 receive | I bit | CAN2RIER (RXFIE) | No | Yes |
| Vector base + $A0 | $50 | CAN2 transmit | I bit | CAN2TIER (TXEIE[2:0]) | No | Yes |
| Vector base + $9E | $4F | CAN3 wake-up | I bit | CAN3RIER (WUPIE) | Yes | Yes |
| Vector base+ $9C | $4E | CAN3 errors | I bit | CAN3RIER (CSCIE, OVRIE) | No | Yes |
| Vector base+ $9A | $4D | CAN3 receive | I bit | CAN3RIER (RXFIE) | No | Yes |
| Vector base + $98 | $4C | CAN3 transmit | I bit | CAN3TIER (TXEIE[2:0]) | No | Yes |
| Vector base + $96 | $4B | CAN4 wake-up | I bit | CAN4RIER (WUPIE) | Yes | Yes |
| Vector base + $94 | $4A | CAN4 errors | I bit | CAN4RIER (CSCIE, OVRIE) | No | Yes |
| Vector base + $92 | $49 | CAN4 receive | I bit | CAN4RIER (RXFIE) | No | Yes |
| Vector base + $90 | $48 | CAN4 transmit | I bit | CAN4TIER (TXEIE[2:0]) | No | Yes |
| Vector base + $8E | $47 | Port P Interrupt | I bit | PIEP (PIEP7-PIEP0) | Yes | Yes |
| Vector base+ $8C | $46 | PWM emergency shutdown | I bit | PWMSDN (PWMIE) | No | Yes |

**Table 1-14. Interrupt Vector Locations (Sheet 3 of 4)**

| Vector Address[1] | XGATE Channel ID[2] | Interrupt Source | CCR Mask | Local Enable | STOP Wake up | WAIT Wake up |
|---|---|---|---|---|---|---|
| Vector base + $8A | $45 | SCI2 | I bit | SCI2CR2 (TIE, TCIE, RIE, ILIE) | Yes | Yes |
| Vector base + $88 | $44 | SCI3 | I bit | SCI3CR2 (TIE, TCIE, RIE, ILIE) | Yes | Yes |
| Vector base + $86 | $43 | SCI4 | I bit | SCI4CR2 (TIE, TCIE, RIE, ILIE) | Yes | Yes |
| Vector base + $84 | $42 | SCI5 | I bit | SCI5CR2 (TIE, TCIE, RIE, ILIE) | Yes | Yes |
| Vector base + $82 | $41 | IIC1 Bus | I bit | IBCR (IBIE) | No | Yes |
| Vector base + $80 | $40 | Low-voltage interrupt (LVI) | I bit | VREGCTRL (LVIE) | No | Yes |
| Vector base + $7E | $3F | Autonomous periodical interrupt (API) | I bit | VREGAPICTRL (APIE) | Yes | Yes |
| Vector base + $7C | — | High Temperature Interrupt | I bit | VREGHTCL (HTIE) | No | Yes |
| Vector base + $7A | $3D | Periodic interrupt timer channel 0 | I bit | PITINTE (PINTE0) | No | Yes |
| Vector base + $78 | $3C | Periodic interrupt timer channel 1 | I bit | PITINTE (PINTE1) | No | Yes |
| Vector base + $76 | $3B | Periodic interrupt timer channel 2 | I bit | PITINTE (PINTE2) | No | Yes |
| Vector base + $74 | $3A | Periodic interrupt timer channel 3 | I bit | PITINTE (PINTE3) | No | Yes |
| Vector base + $72 | $39 | XGATE software trigger 0 | I bit | XGMCTL (XGIE) | No | Yes |
| Vector base + $70 | $38 | XGATE software trigger 1 | I bit | XGMCTL (XGIE) | No | Yes |
| Vector base + $6E | $37 | XGATE software trigger 2 | I bit | XGMCTL (XGIE) | No | Yes |
| Vector base + $6C | $36 | XGATE software trigger 3 | I bit | XGMCTL (XGIE) | No | Yes |
| Vector base + $6A | $35 | XGATE software trigger 4 | I bit | XGMCTL (XGIE) | No | Yes |
| Vector base + $68 | $34 | XGATE software trigger 5 | I bit | XGMCTL (XGIE) | No | Yes |
| Vector base + $66 | $33 | XGATE software trigger 6 | I bit | XGMCTL (XGIE) | No | Yes |
| Vector base + $64 | $32 | XGATE software trigger 7 | I bit | XGMCTL (XGIE) | No | Yes |
| Vector base + $62 | Reserved | | | | | |
| Vector base + $60 | Reserved | | | | | |
| Vector base + $5E | $2F | Periodic interrupt timer channel 4 | I bit | PITINTE (PINTE4) | No | Yes |
| Vector base + $5C | $2E | Periodic interrupt timer channel 5 | I bit | PITINTE (PINTE5) | No | Yes |
| Vector base + $5A | $2D | Periodic interrupt timer channel 6 | I bit | PITINTE (PINTE6) | No | Yes |
| Vector base + $58 | $2C | Periodic interrupt timer channel 7 | I bit | PITINTE (PINTE7) | No | Yes |
| Vector base + $56 | $2B | SCI7 | I bit | SCI7CR2 (TIE, TCIE, RIE, ILIE) | Yes | Yes |
| Vector base + $54 | $2A | TIM timer channel 0 | I bit | TIE (C0I) | No | Yes |
| Vector base + $52 | $29 | TIM timer channel 1 | I bit | TIE (C1I) | No | Yes |
| Vector base + $50 | $28 | TIM timer channel 2 | I bit | TIE (C2I) | No | Yes |

**Table 1-14. Interrupt Vector Locations (Sheet 4 of 4)**

| Vector Address[1] | XGATE Channel ID[2] | Interrupt Source | CCR Mask | Local Enable | STOP Wake up | WAIT Wake up |
|---|---|---|---|---|---|---|
| Vector base+ $4E | $27 | TIM timer channel 3 | I bit | TIE (C3I) | No | Yes |
| Vector base + $4C | $26 | TIM timer channel 4 | I bit | TIE (C4I) | No | Yes |
| Vector base+ $4A | $25 | TIM timer channel 5 | I bit | TIE (C5I) | No | Yes |
| Vector base+ $48 | $24 | TIM timer channel 6 | I bit | TIE (C6I) | No | Yes |
| Vector base+ $46 | $23 | TIM timer channel 7 | I bit | TIE (C7I) | No | Yes |
| Vector base+ $44 | $22 | TIM timer overflow | I bit | TSRC2 (TOF) | No | Yes |
| Vector base + $42 | $21 | TIM Pulse accumulator A overflow | I bit | PACTL (PAOVI) | No | Yes |
| Vector base+ $40 | $20 | TIM Pulse accumulator input edge | I bit | PACTL (PAI) | No | Yes |
| Vector base + $3E | $1F | ATD0 Compare Interrupt | I bit | ATD0CTL2 (ACMPIE) | Yes | Yes |
| Vector base + $3C | $1E | ATD1 Compare Interrupt | I bit | ATD1CTL2 (ACMPIE) | Yes | Yes |
| Vector base+ $18 to Vector base + $3A | Reserved | | | | | |
| Vector base + $16 | — | XGATE software error interrupt | None | None | No | Yes |
| Vector base + $14 | — | MPU Access Error | None | None | No | No |
| Vector base + $12 | — | System Call Interrupt (SYS) | — | None | — | — |
| Vector base + $10 | — | Spurious interrupt | — | None | — | — |

1. 16 bits vector address based
2. For detailed description of XGATE channel ID refer to XGATE Block Guide

## 1.6.3 Effects of Reset

When a reset occurs, MCU registers and control bits are initialized. Refer to the respective block descriptions for register reset states.

On each reset, the Flash module executes a reset sequence to load Flash configuration registers and initialize the buffer RAM EEE partition, if required.

### 1.6.3.1 Flash Configuration Reset Sequence (Core Hold Phase)

On each reset, the Flash module will hold CPU activity while loading Flash module registers and configuration from the Flash memory. The duration of this phase is given as $t_{RST}$ in the device electrical parameter specification. If double faults are detected in the reset phase, Flash module protection and security may be active on leaving reset. This is explained in more detail in the Flash module section.

### 1.6.3.2 EEE Reset Sequence Phase (Core Active Phase)

During this phase of the reset sequence (following on from the core hold phase) the CPU can execute instructions while the FTM initialization completes and, if configured for EEE operation, the EEE RAM

is loaded with valid data from the D-Flash EEE partition. Completion of this phase is indicated by the CCIF flag in the FTM FSTAT register becoming set. If the CPU accesses any EEE RAM location before the CCIF flag is set, the CPU is stalled until the FTM reset sequence is complete and the EEE RAM data is valid. Once the CCIF flag is set, indicating the end of this phase, the EEE RAM can be accessed without impacting the CPU and FTM commands can be executed.

#### 1.6.3.3 Reset While Flash Command Active

If a reset occurs while any Flash command is in progress, that command will be immediately aborted. The state of the word being programmed or the sector/block being erased is not guaranteed.

#### 1.6.3.4 I/O Pins

Refer to the PIM block description for reset configurations of all peripheral module ports.

#### 1.6.3.5 Memory

The RAM arrays are not initialized out of reset.

#### 1.6.3.6 COP Configuration

The COP timeout rate bits CR[2:0] and the WCOP bit in the COPCTL register are loaded on rising edge of $\overline{\text{RESET}}$ from the Flash register FOPT. See Table 1-15 and Table 1-16 for coding. The FOPT register is loaded from the Flash configuration field byte at global address $7FFF0E during the reset sequence.

If the MCU is secured the COP timeout rate is always set to the longest period (CR[2:0] = 111) after COP reset.

**Table 1-15. Initial COP Rate Configuration**

| NV[2:0] in FOPT Register | CR[2:0] in COPCTL Register |
|---|---|
| 000 | 111 |
| 001 | 110 |
| 010 | 101 |
| 011 | 100 |
| 100 | 011 |
| 101 | 010 |
| 110 | 001 |
| 111 | 000 |

**Table 1-16. Initial WCOP Configuration**

| NV[3] in FOPT Register | WCOP in COPCTL Register |
|---|---|
| 1 | 0 |
| 0 | 1 |

## 1.7 ADC0 Configuration

### 1.7.1 External Trigger Input Connection

The ADC module includes four external trigger inputs ETRIG0, ETRIG1, ETRIG2, and ETRIG3. The external trigger allows the user to synchronize ADC conversion to external trigger events. Table 1-17 shows the connection of the external trigger inputs.

**Table 1-17. ATD0 External Trigger Sources**

| External Trigger Input | Connectivity |
|---|---|
| ETRIG0 | Pulse width modulator channel 1 |
| ETRIG1 | Pulse width modulator channel 3 |
| ETRIG2 | Periodic interrupt timer hardware trigger 0 |
| ETRIG3 | Periodic interrupt timer hardware trigger 1 |

Consult the ATD block description for information about the analog-to-digital converter module. ATD block description refererences to freeze mode are equivalent to active BDM mode.

### 1.7.2 ADC0 Channel[17] Connection

Further to the 16 externally available channels, ADC0 features an extra channel[17] that is connected to the internal temperature sensor at device level. To access this channel ADC0 must use the channel encoding SC:CD:CC:CB:CA = 1:0:0:0:1 in ATDCTL5. For more temperature sensor information, please refer to 1.10.1 Temperature Sensor Configuration

## 1.8 ADC1 External Trigger Input Connection

The ADC module includes four external trigger inputs ETRIG0, ETRIG1, ETRIG2, and ETRIG3. The external trigger feature allows the user to synchronize ADC conversion to external trigger events. Table 1-18 shows the connection of the external trigger inputs.

**Table 1-18. ATD1 External Trigger Sources**

| External Trigger Input | Connectivity |
|---|---|
| ETRIG0 | Pulse width modulator channel 1 |
| ETRIG1 | Pulse width modulator channel 3 |
| ETRIG2 | Periodic interrupt timer hardware trigger 0 |
| ETRIG3 | Periodic interrupt timer hardware trigger 1 |

Consult the ADC block description for information about the analog-to-digital converter module. ADC block description refererences to freeze mode are equivalent to active BDM mode.

## 1.9 MPU Configuration

The MPU has the option of a third bus master (CPU + XGATE + other) which is not present on this device family but may be on other parts.

## 1.10 VREG Configuration

The VREGEN connection of the voltage regulator is tied internally to VDDR such that the voltage regulator is always enabled with VDDR connected to a positive supply voltage. The device must be configured with the internal voltage regulator enabled. Operation in conjunction with an external voltage regulator is not supported.

The autonomous periodic interrupt clock output is mapped to PortT[5].

The API trimming register APITR is loaded on rising edge of $\overline{\text{RESET}}$ from the Flash IFR option field at global address 0x40_00F0 bits[5:0] during the reset sequence. Currently factory programming of this IFR range is not supported.

### 1.10.1 Temperature Sensor Configuration

The VREG high temperature trimming register bits VREGHTTR[3:0] are loaded from the internal Flash during the reset sequence. To use the high temperature interrupt within the specified limits ($T_{HTIA}$ and $T_{HTID}$) these bits must be loaded with 0x8. Currently factory programming  is not supported.

The device temperature can be monitored on ADC0 channel[17].
The internal bandgap reference voltage can also be mapped to ADC0 analog input channel[17]. The voltage regulator VSEL bit when set, maps the bandgap and, when clear, maps the temperature sensor to ADC0 channel[17].

Read access to reserved VREG register space returns “0”. Write accesses have no effect. This device does not support access abort of reserved VREG register space.

## 1.11 BDM Clock Configuration

The BDM alternate clock source is the oscillator clock.

## 1.12 S12XEPIM Configuration

On smaller derivatives the S12XEPIM module is a subset of the S12XEP100. The registers of the unavailable ports are unimplemented.

## 1.13 Oscillator Configuration

The $\overline{\text{XCLKS}}$ is an input signal which controls whether a crystal in combination with the internal loop controlled (low power) Pierce oscillator is used or whether full swing Pierce oscillator/external clock circuitry is used. For this device $\overline{\text{XCLKS}}$ is mapped to PE7.

The $\overline{\text{XCLKS}}$ signal selects the oscillator configuration during reset low phase while a clock quality check is ongoing. This is the case for:

- Power on reset or low-voltage reset
- Clock monitor reset
- Any reset while in self-clock mode or full stop mode

The selected oscillator configuration is frozen with the rising edge of the RESET pin in any of these above described reset cases.

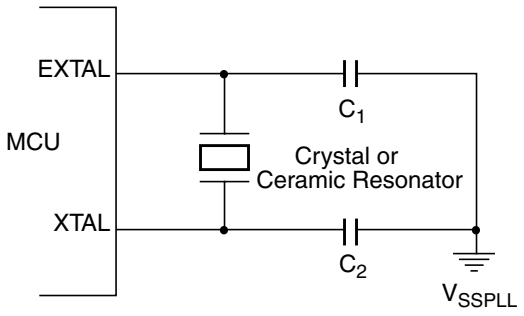



**Figure 1-10. Loop Controlled Pierce Oscillator Connections ($\overline{\text{XCLKS}}$ = 1)**

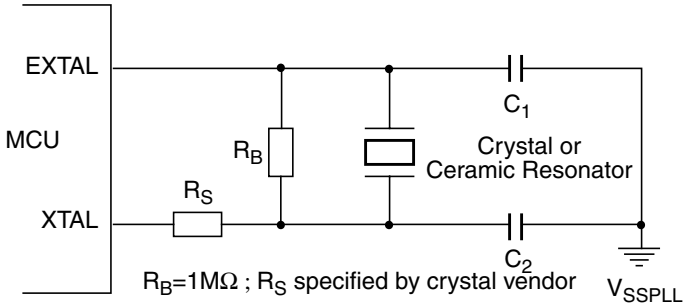



**Figure 1-11. Full Swing Pierce Oscillator Connections ($\overline{\text{XCLKS}}$ = 0)**

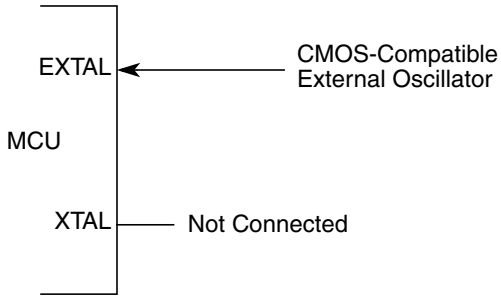



**Figure 1-12. External Clock Connections ($\overline{\text{XCLKS}}$ = 0)**

# Chapter 2
# Port Integration Module (S12XEPIMV1)

**Table 2-1. Revision History**

| Revision Number | Revision Date | Sections Affected | Description of Changes |
|---|---|---|---|
| V01.17 | 02 Apr 2008 | | • Corrected reduced drive strength to 1/5<br>• Separated PE1,0 bit descriptions from other PE GPIO |
| V01.18 | 25 Nov 2008 | 2.3.19/120<br>2.4.3.4/181 | • Corrected alternative functions on Port K (ACC[2:0])<br>• Corrected functions on PE[5] (MODB) and PE[2] (WE) |
| V01.19 | 18 Dec 2009 | | • Added function independency to reduced drive and wired-or bit descriptions<br>• Minor corrections |

## 2.1 Introduction

### 2.1.1 Overview

The S12XE Family Port Integration Module establishes the interface between the peripheral modules including the non-multiplexed External Bus Interface module (S12X_EBI) and the I/O pins for all ports. It controls the electrical pin properties as well as the signal prioritization and multiplexing on shared pins.

This document covers:

- Port A and B used as address output of the S12X_EBI
- Port C and D used as data I/O of the S12X_EBI
- Port E associated with the S12X_EBI control signals and the $\overline{\text{IRQ}}$, $\overline{\text{XIRQ}}$ interrupt inputs
- Port K associated with address output and control signals of the S12X_EBI
- Port T associated with 1 ECT module
- Port S associated with 2 SCI and 1 SPI modules
- Port M associated with 4 MSCAN and 1 SCI module
- Port P connected to the PWM and 2 SPI modules - inputs can be used as an external interrupt source
- Port H associated with 4 SCI modules - inputs can be used as an external interrupt source
- Port J associated with 1 MSCAN, 1 SCI, 2 IIC modules and chip select outputs - inputs can be used as an external interrupt source
- Port AD0 and AD1 associated with two 16-channel ATD modules
- Port R associated with 1 standard timer (TIM) module
- Port L associated with 4 SCI modules

- Port F associated with IIC, SCI and chip select outputs

Most I/O pins can be configured by register bits to select data direction and drive strength, to enable and select pull-up or pull-down devices.

**NOTE**

This document assumes the availability of all features (208-pin package option). Some functions are not available on lower pin count package options. Refer to the pin-out summary in the SOC Guide.

### 2.1.2 Features

The Port Integration Module includes these distinctive registers:

- Data and data direction registers for Ports A, B, C, D, E, K, T, S, M, P, H, J, AD0, AD1, R, L, and F when used as general-purpose I/O
- Control registers to enable/disable pull-device and select pull-ups/pull-downs on Ports T, S, M, P, H, J, R, L, and F on per-pin basis
- Control registers to enable/disable pull-up devices on Ports AD0 and AD1 on per-pin basis
- Single control register to enable/disable pull-ups on Ports A, B, C, D, E, and K on per-port basis and on BKGD pin
- Control registers to enable/disable reduced output drive on Ports T, S, M, P, H, J, AD0, AD1, R, L, and F on per-pin basis
- Single control register to enable/disable reduced output drive on Ports A, B, C, D, E, and K on per-port basis
- Control registers to enable/disable open-drain (wired-or) mode on Ports S, M, and L
- Interrupt flag register for pin interrupts on Ports P, H, and J
- Control register to configure $\overline{\text{IRQ}}$ pin operation
- Free-running clock outputs

A standard port pin has the following minimum features:

- Input/output selection
- 5V output drive with two selectable drive strengths
- 5V digital and analog input
- Input with selectable pull-up or pull-down device

Optional features supported on dedicated pins:

- Open drain for wired-or connections
- Interrupt inputs with glitch filtering
- Reduced input threshold to support low voltage applications

## 2.2 External Signal Description

This section lists and describes the signals that do connect off-chip.

Table 2-2 shows all the pins and their functions that are controlled by the Port Integration Module. *Refer to the SOC Guide for the availability of the individual pins in the different package options.*

**NOTE**

If there is more than one function associated with a pin, the priority is indicated by the position in the table from top (highest priority) to bottom (lowest priority).

**Table 2-2. Pin Functions and Priorities**

| Port | Pin Name | Pin Function & Priority[1] | I/O | Description | Pin Function after Reset |
|---|---|---|---|---|---|
| - | BKGD | MODC [2] | I | MODC input during $\overline{\text{RESET}}$ | BKGD |
| | | BKGD | I/O | S12X_BDM communication pin | |
| A | PA[7:0] | ADDR[15:8] mux IVD[15:8] [3] | O | High-order external bus address output (multiplexed with IVIS data) | Mode dependent [4] |
| | | GPIO | I/O | General-purpose I/O | |
| B | PB[7:1] | ADDR[7:1] mux IVD[7:1] [3] | O | Low-order external bus address output (multiplexed with IVIS data) | Mode dependent [4] |
| | | GPIO | I/O | General-purpose I/O | |
| | PB[0] | ADDR[0] mux IVD0 [3] | O | Low-order external bus address output (multiplexed with IVIS data) | |
| | | $\overline{\text{UDS}}$ | O | Upper data strobe | |
| | | GPIO | I/O | General-purpose I/O | |
| C | PC[7:0] | DATA[15:8] | I/O | High-order bidirectional data input/output Configurable for reduced input threshold | Mode dependent [4] |
| | | GPIO | I/O | General-purpose I/O | |
| D | PD[7:0] | DATA[7:0] | I/O | Low-order bidirectional data input/output Configurable for reduced input threshold | Mode dependent [4] |
| | | GPIO | I/O | General-purpose I/O | |

| Port | Pin Name | Pin Function & Priority[1] | I/O | Description | Pin Function after Reset |
|---|---|---|---|---|---|
| E | PE[7] | $\overline{\text{XCLKS}}$ [2] | I | External clock selection input during $\overline{\text{RESET}}$ | Mode dependent [4] |
| | | ECLKX2 | I | Free-running clock output at Core Clock rate (ECLK x 2) | |
| | | GPIO | I/O | General-purpose I/O | |
| | PE[6] | MODB [2] | I | MODB input during $\overline{\text{RESET}}$ | |
| | | $\overline{\text{TAGHI}}$ | I | Instruction tagging low pin<br>Configurable for reduced input threshold | |
| | | GPIO | I/O | General-purpose I/O | |
| | PE[5] | MODA [2] | I | MODA input during $\overline{\text{RESET}}$ | |
| | | $\overline{\text{RE}}$ | O | Read enable signal | |
| | | $\overline{\text{TAGLO}}$ | I | Instruction tagging low pin<br>Configurable for reduced input threshold | |
| | | GPIO | I/O | General-purpose I/O | |
| | PE[4] | ECLK | O | Free-running clock output at the Bus Clock rate or programmable divided in normal modes | |
| | | GPIO | I/O | General-purpose I/O | |
| | PE[3] | EROMCTL [2] | I | EROMON bit control input during $\overline{\text{RESET}}$ | |
| | | $\overline{\text{LSTRB}}$ | O | Low strobe bar output | |
| | | $\overline{\text{LDS}}$ | O | Lower data strobe | |
| | | GPIO | I/O | General-purpose I/O | |
| | PE[2] | R$\overline{\text{W}}$ | O | Read/write output for external bus | |
| | | $\overline{\text{WE}}$ | O | Write enable signal | |
| | | GPIO | I/O | General-purpose I/O | |
| | PE[1] | $\overline{\text{IRQ}}$ | I | Maskable level- or falling edge-sensitive interrupt input | |
| | | GPI | I | General-purpose input | |
| | PE[0] | $\overline{\text{XIRQ}}$ | I | Non-maskable level-sensitive interrupt input | |
| | | GPI | I | General-purpose input | |
| K | PK[7] | ROMCTL [2] | I | ROMON bit control input during $\overline{\text{RESET}}$ | Mode dependent [3] |
| | | $\overline{\text{EWAIT}}$ | I | External Wait signal<br>Configurable for reduced input threshold | |
| | | GPIO | I/O | General-purpose I/O | |
| | PK[6:4] | ADDR[22:20] mux ACC[2:0] [3] | O | Extended external bus address output (multiplexed with access master output) | |
| | | GPIO | I/O | General-purpose I/O | |
| | PK[3:0] | ADDR[19:16] mux IQSTAT[3:0] [3] | O | Extended external bus address output (multiplexed with instruction pipe status bits) | |
| | | GPIO | I/O | General-purpose I/O | |

| Port | Pin Name | Pin Function & Priority[1] | I/O | Description | Pin Function after Reset |
|---|---|---|---|---|---|
| T | PT[7] | IOC[7] | I/O | Enhanced Capture Timer Channels 7 input/output | GPIO |
| | | GPIO | I/O | General-purpose I/O | |
| | PT[5] | IOC[5] | I/O | Enhanced Capture Timer Channel 5 input/output | |
| | | VREG_API | O | VREG Autonomous Periodical Interrupt output | |
| | | GPIO | I/O | General-purpose I/O | |
| | PT[4:0] | IOC[4:0] | I/O | Enhanced Capture Timer Channels 4 - 0 input/output | |
| | | GPIO | I/O | General-purpose I/O | |
| S | PS7 | $\overline{SS0}$ | I/O | Serial Peripheral Interface 0 slave select output in master mode, input in slave mode or master mode. | GPIO |
| | | GPIO | I/O | General-purpose I/O | |
| | PS6 | SCK0 | I/O | Serial Peripheral Interface 0 serial clock pin | |
| | | GPIO | I/O | General-purpose I/O | |
| | PS5 | MOSI0 | I/O | Serial Peripheral Interface 0 master out/slave in pin | |
| | | GPIO | I/O | General-purpose I/O | |
| | PS4 | MISO0 | I/O | Serial Peripheral Interface 0 master in/slave out pin | |
| | | GPIO | I/O | General-purpose I/O | |
| | PS3 | TXD1 | O | Serial Communication Interface 1 transmit pin | |
| | | GPIO | I/O | General-purpose I/O | |
| | PS2 | RXD1 | I | Serial Communication Interface 1 receive pin | |
| | | GPIO | I/O | General-purpose I/O | |
| | PS1 | TXD0 | O | Serial Communication Interface 0 transmit pin | |
| | | GPIO | I/O | General-purpose I/O | |
| | PS0 | RXD0 | I | Serial Communication Interface 0 receive pin | |
| | | GPIO | I/O | General-purpose I/O | |

| Port | Pin Name | Pin Function & Priority[1] | I/O | Description | Pin Function after Reset |
|---|---|---|---|---|---|
| M | PM7 | TXCAN3 | O | MSCAN3 transmit pin | GPIO |
| | | (TXCAN4) | O | MSCAN4 transmit pin | |
| | | TXD3 | O | Serial Communication Interface 3 transmit pin | |
| | | GPIO | I/O | General-purpose I/O | |
| | PM6 | RXCAN3 | I | MSCAN3 receive pin | |
| | | (RXCAN4) | I | MSCAN4 receive pin | |
| | | RXD3 | I | Serial Communication Interface 3 receive pin | |
| | | GPIO | I/O | General-purpose I/O | |
| | PM5 | TXCAN2 | O | MSCAN2 transmit pin | |
| | | (TXCAN0) | O | MSCAN0 transmit pin | |
| | | (TXCAN4) | O | MSCAN4 transmit pin | |
| | | (SCK0) | I/O | Serial Peripheral Interface 0 serial clock pin<br>If CAN0 is routed to PM[3:2] the SPI0 can still be used in bidirectional master mode. | |
| | | GPIO | I/O | General-purpose I/O | |
| | PM4 | RXCAN2 | I | MSCAN2 receive pin | |
| | | (RXCAN0) | I | MSCAN0 receive pin | |
| | | (RXCAN4) | I | MSCAN4 receive pin | |
| | | (MOSI0) | I/O | Serial Peripheral Interface 0 master out/slave in pin<br>If CAN0 is routed to PM[3:2] the SPI0 can still be used in bidirectional master mode. | |
| | | GPIO | I/O | General-purpose I/O | |
| | PM3 | TXCAN1 | O | MSCAN1 transmit pin | |
| | | (TXCAN0) | O | MSCAN0 transmit pin | |
| | | ($\overline{SS0}$) | I/O | Serial Peripheral Interface 0 slave select output in master mode, input for slave mode or master mode. | |
| | | GPIO | I/O | General-purpose I/O | |
| | PM2 | RXCAN1 | I | MSCAN1 receive pin | |
| | | (RXCAN0) | I | MSCAN0 receive pin | |
| | | (MISO0) | I/O | Serial Peripheral Interface 0 master in/slave out pin | |
| | | GPIO | I/O | General-purpose I/O | |
| | PM1 | TXCAN0 | O | MSCAN0 transmit pin | |
| | | GPIO | I/O | General-purpose I/O | |
| | PM0 | RXCAN0 | I | MSCAN0 receive pin | |
| | | GPIO | I/O | General-purpose I/O | |

| Port | Pin Name | Pin Function & Priority[1] | I/O | Description | Pin Function after Reset |
|---|---|---|---|---|---|
| P | PP7 | PWM7 | I/O | Pulse Width Modulator input/output channel 7 | GPIO |
| | | SCK2 | I/O | Serial Peripheral Interface 2 serial clock pin | |
| | | (TIMIOC7) | I/O | Timer Channel 7 input/output | |
| | | GPIO/KWP7 | I/O | General-purpose I/O with interrupt | |
| | PP6 | PWM6 | O | Pulse Width Modulator output channel 6 | |
| | | $\overline{SS2}$ | I/O | Serial Peripheral Interface 2 slave select output in master mode, input for slave mode or master mode. | |
| | | (TIMIOC6) | I/O | Timer Channel 6 input/output | |
| | | GPIO/KWP6 | I/O | General-purpose I/O with interrupt | |
| | PP5 | PWM5 | O | Pulse Width Modulator output channel 5 | |
| | | MOSI2 | I/O | Serial Peripheral Interface 2 master out/slave in pin | |
| | | (TIMIOC5) | I/O | Timer Channel 5 input/output | |
| | | GPIO/KWP5 | I/O | General-purpose I/O with interrupt | |
| | PP4 | PWM4 | O | Pulse Width Modulator output channel 4 | |
| | | MISO2 | I/O | Serial Peripheral Interface 2 master in/slave out pin | |
| | | (TIMIOC4) | I/O | Timer Channel 4 input/output | |
| | | GPIO/KWP4 | I/O | General-purpose I/O with interrupt | |
| | PP3 | PWM3 | O | Pulse Width Modulator output channel 3 | |
| | | $\overline{SS1}$ | I/O | Serial Peripheral Interface 1 slave select output in master mode, input for slave mode or master mode. | |
| | | (TIMIOC3) | I/O | Timer Channel 3 input/output | |
| | | GPIO/KWP3 | I/O | General-purpose I/O with interrupt | |
| | PP2 | PWM2 | O | Pulse Width Modulator output channel 2 | |
| | | SCK1 | I/O | Serial Peripheral Interface 1 serial clock pin | |
| | | (TIMIOC2) | I/O | Timer Channel 2 input/output | |
| | | GPIO/KWP2 | I/O | General-purpose I/O with interrupt | |
| | PP1 | PWM1 | O | Pulse Width Modulator output channel 1 | |
| | | MOSI1 | I/O | Serial Peripheral Interface 1 master out/slave in pin | |
| | | (TIMIOC1) | I/O | Timer Channel 1 input/output | |
| | | GPIO/KWP1 | I/O | General-purpose I/O with interrupt | |
| | PP0 | PWM0 | O | Pulse Width Modulator output channel 0 | |
| | | MISO1 | I/O | Serial Peripheral Interface 1 master in/slave out pin | |
| | | (TIMIOC0) | I/O | Timer Channel 0 input/output | |
| | | GPIO/KWP0 | I/O | General-purpose I/O with interrupt | |

| Port | Pin Name | Pin Function & Priority[1] | I/O | Description | Pin Function after Reset |
|---|---|---|---|---|---|
| H | PH7 | ($\overline{SS2}$) | I/O | Serial Peripheral Interface 2 slave select output in master mode, input for slave mode or master mode | GPIO |
| | | TXD5 | O | Serial Communication Interface 5 transmit pin | |
| | | GPIO/KWH7 | I/O | General-purpose I/O with interrupt | |
| | PH6 | (SCK2) | I/O | Serial Peripheral Interface 2 serial clock pin | |
| | | RXD5 | I | Serial Communication Interface 5 receive pin | |
| | | GPIO/KWH6 | I/O | General-purpose I/O with interrupt | |
| | PH5 | (MOSI2) | I/O | Serial Peripheral Interface 2 master out/slave in pin | |
| | | TXD4 | O | Serial Communication Interface 4 transmit pin | |
| | | GPIO/KWH5 | I/O | General-purpose I/O with interrupt | |
| | PH4 | (MISO2) | I/O | Serial Peripheral Interface 2 master in/slave out pin | |
| | | RXD4 | I | Serial Communication Interface 4 receive pin | |
| | | GPIO/KWH4 | I/O | General-purpose I/O with interrupt | |
| | PH3 | ($\overline{SS1}$) | I/O | Serial Peripheral Interface 1 slave select output in master mode, input for slave mode or master mode. | |
| | | TXD7 | O | Serial Communication Interface 7 transmit pin | |
| | | GPIO/KWH3 | I/O | General-purpose I/O with interrupt | |
| | PH2 | (SCK1) | I/O | Serial Peripheral Interface 1 serial clock pin | |
| | | RXD7 | I | Serial Communication Interface 7 receive pin | |
| | | GPIO/KWH2 | I/O | General-purpose I/O with interrupt | |
| | PH1 | (MOSI1) | I/O | Serial Peripheral Interface 1 master out/slave in pin | |
| | | TXD6 | O | Serial Communication Interface 6 transmit pin | |
| | | GPIO/KWH1 | I/O | General-purpose I/O with interrupt | |
| | PH0 | (MISO1) | I/O | Serial Peripheral Interface 1 master in/slave out pin | |
| | | TXD6 | O | Serial Communication Interface 6 transmit pin | |
| | | GPIO/KWH0 | I/O | General-purpose I/O with interrupt | |

| Port | Pin Name | Pin Function & Priority[1] | I/O | Description | Pin Function after Reset |
|---|---|---|---|---|---|
| J | PJ7 | TXCAN4 | O | MSCAN4 transmit pin | GPIO |
| | | SCL0 | O | Inter Integrated Circuit 0 serial clock line | |
| | | (TXCAN0) | O | MSCAN0 transmit pin | |
| | | GPIO/KWJ7 | I/O | General-purpose I/O with interrupt | |
| | PJ6 | RXCAN4 | I | MSCAN4 receive pin | |
| | | SDA0 | I/O | Inter Integrated Circuit 0 serial data line | |
| | | (RXCAN0) | I | MSCAN0 receive pin | |
| | | GPIO/KWJ6 | I/O | General-purpose I/O with interrupt | |
| | PJ5 | SCL1 | O | Inter Integrated Circuit 1 serial clock line | |
| | | $\overline{CS2}$ | O | Chip select 2 | |
| | | GPIO/KWJ5 | I/O | General-purpose I/O with interrupt | |
| | PJ4 | SDA1 | I/O | Inter Integrated Circuit 1 serial data line | |
| | | $\overline{CS0}$ | O | Chip select 0 | |
| | | GPIO/KWJ4 | I/O | General-purpose I/O with interrupt | |
| | PJ3 | GPIO/KWJ3 | I/O | General-purpose I/O with interrupt | |
| | PJ2 | $\overline{CS1}$ | O | Chip select 1 | |
| | | GPIO/KWJ2 | I/O | General-purpose I/O with interrupt | |
| | PJ1 | TXD2 | O | Serial Communication Interface 2 transmit pin | |
| | | GPIO/KWJ1 | I/O | General-purpose I/O with interrupt | |
| | PJ0 | RXD2 | I | Serial Communication Interface 2 receive pin | |
| | | $\overline{CS3}$ | O | Chip select 3 | |
| | | GPIO/KWJ0 | I/O | General-purpose I/O with interrupt | |
| AD0 | PAD[15:0] | GPIO | I/O | General-purpose I/O | GPIO |
| | | AN[15:0] | I | ATD0 analog inputs | |
| AD1 | PAD[31:16] | GPIO | I/O | General-purpose I/O | GPIO |
| | | AN[15:0] | I | ATD1 analog inputs | |
| R | PR[7:0] | TIMIOC[7:0] | I/O | Timer Channels 7- 0 input/output | GPIO |
| | | GPIO | I/O | General-purpose I/O | |

| Port | Pin Name | Pin Function & Priority[1] | I/O | Description | Pin Function after Reset |
|---|---|---|---|---|---|
| L | PL7 | (TXD7) | O | Serial Communication Interface 7 transmit pin | GPIO |
| | | GPIO | I/O | General-purpose I/O | |
| | PL6 | (RXD7) | I | Serial Communication Interface 7 receive pin | |
| | | GPIO | I/O | General-purpose I/O | |
| | PL5 | (TXD6) | O | Serial Communication Interface 6 transmit pin | |
| | | GPIO | I/O | General-purpose I/O | |
| | PL4 | (RXD6) | I | Serial Communication Interface 6 receive pin | |
| | | GPIO | I/O | General-purpose I/O | |
| | PL3 | (TXD5) | O | Serial Communication Interface 5 transmit pin | |
| | | GPIO | I/O | General-purpose I/O | |
| | PL2 | (RXD5) | I | Serial Communication Interface 5 receive pin | |
| | | GPIO | I/O | General-purpose I/O | |
| | PL1 | (TXD4) | O | Serial Communication Interface 4 transmit pin | |
| | | GPIO | I/O | General-purpose I/O | |
| | PL0 | (RXD4) | I | Serial Communication Interface 4 receive pin | |
| | | GPIO | I/O | General-purpose I/O | |
| F | PF7 | (TXD3) | O | Serial Communication Interface 3 transmit pin | GPIO |
| | | GPIO | I/O | General-purpose I/O | |
| | PF6 | (RXD3) | I | Serial Communication Interface 3 receive pin | |
| | | GPIO | I/O | General-purpose I/O | |
| | PF5 | (SCL0) | O | Inter Integrated Circuit 0 serial clock line | |
| | | GPIO | I/O | General-purpose I/O | |
| | PF4 | (SDA0) | I/O | Inter Integrated Circuit 0 serial data line | |
| | | GPIO | I/O | General-purpose I/O | |
| | PF3 | ($\overline{CS3}$) | O | Chip select 3 | |
| | | GPIO | I/O | General-purpose I/O | |
| | PF2 | ($\overline{CS2}$) | O | Chip select 2 | |
| | | GPIO | I/O | General-purpose I/O | |
| | PF1 | ($\overline{CS1}$) | O | Chip select 1 | |
| | | GPIO | I/O | General-purpose I/O | |
| | PF0 | ($\overline{CS0}$) | O | Chip select 0 | |
| | | GPIO | I/O | General-purpose I/O | |

1. Signals in brackets denote alternative module routing pins.
2. Function active when $\overline{RESET}$ asserted.
3. Only available in emulation modes or in Special Test Mode with IVIS on.
4. Refer to S12X_EBI section.

## 2.3 Memory Map and Register Definition

This section provides a detailed description of all Port Integration Module registers.

## 2.3.1 Memory Map

| Register Name | | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|---|
| 0x0000 PORTA | R | PA7 | PA6 | PA5 | PA4 | PA3 | PA2 | PA1 | PA0 |
| | W | | | | | | | | |
| 0x0001 PORTB | R | PB7 | PB6 | PB5 | PB4 | PB3 | PB2 | PB1 | PB0 |
| | W | | | | | | | | |
| 0x0002 DDRA | R | DDRA7 | DDRA6 | DDRA5 | DDRA4 | DDRA3 | DDRA2 | DDRA1 | DDRA0 |
| | W | | | | | | | | |
| 0x0003 DDRB | R | DDRB7 | DDRB6 | DDRB5 | DDRB4 | DDRB3 | DDRB2 | DDRB1 | DDRB0 |
| | W | | | | | | | | |
| 0x0004 PORTC | R | PC7 | PC6 | PC5 | PC4 | PC3 | PC2 | PC1 | PC0 |
| | W | | | | | | | | |
| 0x0005 PORTD | R | PD7 | PD6 | PD5 | PD4 | PD3 | PD2 | PD1 | PD0 |
| | W | | | | | | | | |
| 0x0006 DDRC | R | DDRC7 | DDRC6 | DDRC5 | DDRC4 | DDRC3 | DDRC2 | DDRC1 | DDRC0 |
| | W | | | | | | | | |
| 0x0007 DDRD | R | DDRD7 | DDRD6 | DDRD5 | DDRD4 | DDRD3 | DDRD2 | DDRD1 | DDRD0 |
| | W | | | | | | | | |
| 0x0008 PORTE | R | PE7 | PE6 | PE5 | PE4 | PE3 | PE2 | PE1 | PE0 |
| | W | | | | | | | | |
| 0x0009 DDRE | R | DDRE7 | DDRE6 | DDRE5 | DDRE4 | DDRE3 | DDRE2 | 0 | 0 |
| | W | | | | | | | | |
| 0x000A 0x000B Non-PIM Address Range | R | Non-PIM Address Range | | | | | | | |
| | W | | | | | | | | |
| 0x000C PUCR | R | PUPKE | BKPUE | 0 | PUPEE | PUPDE | PUPCE | PUPBE | PUPAE |
| | W | | | | | | | | |
| 0x000D RDRIV | R | RDPK | 0 | 0 | RDPE | RDPD | RDPC | RDPB | RDPA |
| | W | | | | | | | | |

= Unimplemented or Reserved

| Register Name | | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|---|
| 0x000E–0x001B Non-PIM Address Range | R<br>W | Non-PIM Address Range | | | | | | | |
| 0x001C ECLKCTL | R | NECLK | NCLKX2 | DIV16 | EDIV4 | EDIV3 | EDIV2 | EDIV1 | EDIV0 |
| | W | | | | | | | | |
| 0x001D Reserved | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | W | | | | | | | | |
| 0x001E IRQCR | R | IRQE | IRQEN | 0 | 0 | 0 | 0 | 0 | 0 |
| | W | | | | | | | | |
| 0x001F Reserved | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | W | | | | | | | | |
| 0x0020–0x0031 Non-PIM Address Range | R<br>W | Non-PIM Address Range | | | | | | | |
| 0x0032 PORTK | R | PK7 | PK6 | PK5 | PK4 | PK3 | PK2 | PK1 | PK0 |
| | W | | | | | | | | |
| 0x0033 DDRK | R | DDRK7 | DDRK6 | DDRK5 | DDRK4 | DDRK3 | DDRK2 | DDRK1 | DDRK0 |
| | W | | | | | | | | |
| 0x0034–0x023F Non-PIM Address Range | R<br>W | Non-PIM Address Range | | | | | | | |
| 0x0240 PTT | R | PTT7 | PTT6 | PTT5 | PTT4 | PTT3 | PTT2 | PTT1 | PTT0 |
| | W | | | | | | | | |
| 0x0241 PTIT | R | PTIT7 | PTIT6 | PTIT5 | PTIT4 | PTIT3 | PTIT2 | PTIT1 | PTIT0 |
| | W | | | | | | | | |
| 0x0242 DDRT | R | DDRT7 | DDRT6 | DDRT5 | DDRT4 | DDRT3 | DDRT2 | DDRT1 | DDRT0 |
| | W | | | | | | | | |
| 0x0243 RDRT | R | RDRT7 | RDRT6 | RDRT5 | RDRT4 | RDRT3 | RDRT2 | RDRT1 | RDRT0 |
| | W | | | | | | | | |

= Unimplemented or Reserved

| Register Name | | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|---|
| 0x0244 PERT | R | PERT7 | PERT6 | PERT5 | PERT4 | PERT3 | PERT2 | PERT1 | PERT0 |
| | W | | | | | | | | |
| 0x0245 PPST | R | PPST7 | PPST6 | PPST5 | PPST4 | PPST3 | PPST2 | PPST1 | PPST0 |
| | W | | | | | | | | |
| 0x0246 Reserved | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | W | | | | | | | | |
| 0x0247 Reserved | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | W | | | | | | | | |
| 0x0248 PTS | R | PTS7 | PTS6 | PTS5 | PTS4 | PTS3 | PTS2 | PTS1 | PTS0 |
| | W | | | | | | | | |
| 0x0249 PTIS | R | PTIS7 | PTIS6 | PTIS5 | PTIS4 | PTIS3 | PTIS2 | PTIS1 | PTIS0 |
| | W | | | | | | | | |
| 0x024A DDRS | R | DDRS7 | DDRS6 | DDRS5 | DDRS4 | DDRS3 | DDRS2 | DDRS1 | DDRS0 |
| | W | | | | | | | | |
| 0x024B RDRS | R | RDRS7 | RDRS6 | RDRS5 | RDRS4 | RDRS3 | RDRS2 | RDRS1 | RDRS0 |
| | W | | | | | | | | |
| 0x024C PERS | R | PERS7 | PERS6 | PERS5 | PERS4 | PERS3 | PERS2 | PERS1 | PERS0 |
| | W | | | | | | | | |
| 0x024D PPSS | R | PPSS7 | PPSS6 | PPSS5 | PPSS4 | PPSS3 | PPSS2 | PPSS1 | PPSS0 |
| | W | | | | | | | | |
| 0x024E WOMS | R | WOMS7 | WOMS6 | WOMS5 | WOMS4 | WOMS3 | WOMS2 | WOMS1 | WOMS0 |
| | W | | | | | | | | |
| 0x024F Reserved | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | W | | | | | | | | |
| 0x0250 PTM | R | PTM7 | PTM6 | PTM5 | PTM4 | PTM3 | PTM2 | PTM1 | PTM0 |
| | W | | | | | | | | |
| 0x0251 PTIM | R | PTIM7 | PTIM6 | PTIM5 | PTIM4 | PTIM3 | PTIM2 | PTIM1 | PTIM0 |
| | W | | | | | | | | |
| 0x0252 DDRM | R | DDRM7 | DDRM6 | DDRM5 | DDRM4 | DDRM3 | DDRM2 | DDRM1 | DDRM0 |
| | W | | | | | | | | |

= Unimplemented or Reserved

| Register Name | | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|---|
| 0x0253 RDRM | R | RDRM7 | RDRM6 | RDRM5 | RDRM4 | RDRM3 | RDRM2 | RDRM1 | RDRM0 |
| | W | | | | | | | | |
| 0x0254 PERM | R | PERM7 | PERM6 | PERM5 | PERM4 | PERM3 | PERM2 | PERM1 | PERM0 |
| | W | | | | | | | | |
| 0x0255 PPSM | R | PPSM7 | PPSM6 | PPSM5 | PPSM4 | PPSM3 | PPSM2 | PPSM1 | PPSM0 |
| | W | | | | | | | | |
| 0x0256 WOMM | R | WOMM7 | WOMM6 | WOMM5 | WOMM4 | WOMM3 | WOMM2 | WOMM1 | WOMM0 |
| | W | | | | | | | | |
| 0x0257 MODRR | R | 0 | MODRR6 | MODRR5 | MODRR4 | MODRR3 | MODRR2 | MODRR1 | MODRR0 |
| | W | | | | | | | | |
| 0x0258 PTP | R | PTP7 | PTP6 | PTP5 | PTP4 | PTP3 | PTP2 | PTP1 | PTP0 |
| | W | | | | | | | | |
| 0x0259 PTIP | R | PTIP7 | PTIP6 | PTIP5 | PTIP4 | PTIP3 | PTIP2 | PTIP1 | PTIP0 |
| | W | | | | | | | | |
| 0x025A DDRP | R | DDRP7 | DDRP6 | DDRP5 | DDRP4 | DDRP3 | DDRP2 | DDRP1 | DDRP0 |
| | W | | | | | | | | |
| 0x025B RDRP | R | RDRP7 | RDRP6 | RDRP5 | RDRP4 | RDRP3 | RDRP2 | RDRP1 | RDRP0 |
| | W | | | | | | | | |
| 0x025C PERP | R | PERP7 | PERP6 | PERP5 | PERP4 | PERP3 | PERP2 | PERP1 | PERP0 |
| | W | | | | | | | | |
| 0x025D PPSP | R | PPSP7 | PPSP6 | PPSP5 | PPSP4 | PPSP3 | PPSP2 | PPSP1 | PPSP0 |
| | W | | | | | | | | |
| 0x025E PIEP | R | PIEP7 | PIEP6 | PIEP5 | PIEP4 | PIEP3 | PIEP2 | PIEP1 | PIEP0 |
| | W | | | | | | | | |
| 0x025F PIFP | R | PIFP7 | PIFP6 | PIFP5 | PIFP4 | PIFP3 | PIFP2 | PIFP1 | PIFP0 |
| | W | | | | | | | | |
| 0x0260 PTH | R | PTH7 | PTH6 | PTH5 | PTH4 | PTH3 | PTH2 | PTH1 | PTH0 |
| | W | | | | | | | | |
| 0x0261 PTIH | R | PTIH7 | PTIH6 | PTIH5 | PTIH4 | PTIH3 | PTIH2 | PTIH1 | PTIH0 |
| | W | | | | | | | | |
| 0x0262 DDRH | R | DDRH7 | DDRH6 | DDRH5 | DDRH4 | DDRH3 | DDRH2 | DDRH1 | DDRH0 |
| | W | | | | | | | | |

= Unimplemented or Reserved

| Register Name | | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|---|
| 0x0263 RDRH | R | RDRH7 | RDRH6 | RDRH5 | RDRH4 | RDRH3 | RDRH2 | RDRH1 | RDRH0 |
| | W | | | | | | | | |
| 0x0264 PERH | R | PERH7 | PERH6 | PERH5 | PERH4 | PERH3 | PERH2 | PERH1 | PERH0 |
| | W | | | | | | | | |
| 0x0265 PPSH | R | PPSH7 | PPSH6 | PPSH5 | PPSH4 | PPSH3 | PPSH2 | PPSH1 | PPSH0 |
| | W | | | | | | | | |
| 0x0266 PIEH | R | PIEH7 | PIEH6 | PIEH5 | PIEH4 | PIEH3 | PIEH2 | PIEH1 | PIEH0 |
| | W | | | | | | | | |
| 0x0267 PIFH | R | PIFH7 | PIFH6 | PIFH5 | PIFH4 | PIFH3 | PIFH2 | PIFH1 | PIFH0 |
| | W | | | | | | | | |
| 0x0268 PTJ | R | PTJ7 | PTJ6 | PTJ5 | PTJ4 | PTJ3 | PTJ2 | PTJ1 | PTJ0 |
| | W | | | | | | | | |
| 0x0269 PTIJ | R | PTIJ7 | PTIJ6 | PTIJ5 | PTIJ4 | PTIJ3 | PTIJ2 | PTIJ1 | PTIJ0 |
| | W | | | | | | | | |
| 0x026A DDRJ | R | DDRJ7 | DDRJ6 | DDRJ5 | DDRJ4 | DDRJ3 | DDRJ2 | DDRJ1 | DDRJ0 |
| | W | | | | | | | | |
| 0x026B RDRJ | R | RDRJ7 | RDRJ6 | RDRJ5 | RDRJ4 | RDRJ3 | RDRJ2 | RDRJ1 | RDRJ0 |
| | W | | | | | | | | |
| 0x026C PERJ | R | PERJ7 | PERJ6 | PERJ5 | PERJ4 | PERJ3 | PERJ2 | PERJ1 | PERJ0 |
| | W | | | | | | | | |
| 0x026D PPSJ | R | PPSJ7 | PPSJ6 | PPSJ5 | PPSJ4 | PPSJ3 | PPSJ2 | PPSJ1 | PPSJ0 |
| | W | | | | | | | | |
| 0x026E PIEJ | R | PIEJ7 | PIEJ6 | PIEJ5 | PIEJ4 | PIEJ3 | PIEJ2 | PIEJ1 | PIEJ0 |
| | W | | | | | | | | |
| 0x026F PIFJ | R | PIFJ7 | PIFJ6 | PIFJ5 | PIFJ4 | PIFJ3 | PIFJ2 | PIFJ1 | PIFJ0 |
| | W | | | | | | | | |
| 0x0270 PT0AD0 | R | PT0AD07 | PT0AD06 | PT0AD05 | PT0AD04 | PT0AD03 | PT0AD02 | PT0AD01 | PT0AD00 |
| | W | | | | | | | | |
| 0x0271 PT1AD0 | R | PT1AD07 | PT1AD06 | PT1AD05 | PT1AD04 | PT1AD03 | PT1AD02 | PT1AD01 | PT1AD00 |
| | W | | | | | | | | |

= Unimplemented or Reserved

| Register Name | | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|---|
| 0x0272 DDR0AD0 | R<br>W | DDR0AD07 | DDR0AD06 | DDR0AD05 | DDR0AD04 | DDR0AD03 | DDR0AD02 | DDR0AD01 | DDR0AD00 |
| 0x0273 DDR1AD0 | R<br>W | DDR1AD07 | DDR1AD06 | DDR1AD05 | DDR1AD04 | DDR1AD03 | DDR1AD02 | DDR1AD01 | DDR1AD00 |
| 0x0274 RDR0AD0 | R<br>W | RDR0AD07 | RDR0AD06 | RDR0AD05 | RDR0AD04 | RDR0AD03 | RDR0AD02 | RDR0AD01 | RDR0AD00 |
| 0x0275 RDR1AD0 | R<br>W | RDR1AD07 | RDR1AD06 | RDR1AD05 | RDR1AD04 | RDR1AD03 | RDR1AD02 | RDR1AD01 | RDR1AD00 |
| 0x0276 PER0AD0 | R<br>W | PER0AD07 | PER0AD06 | PER0AD05 | PER0AD04 | PER0AD03 | PER0AD02 | PER0AD01 | PER0AD00 |
| 0x0277 PER1AD0 | R<br>W | PER1AD07 | PER1AD06 | PER1AD05 | PER1AD04 | PER1AD03 | PER1AD02 | PER1AD01 | PER1AD00 |
| 0x0278 PT0AD1 | R<br>W | PT0AD17 | PT0AD16 | PT0AD15 | PT0AD14 | PT0AD13 | PT0AD12 | PT0AD11 | PT0AD10 |
| 0x0279 PT1AD1 | R<br>W | PT1AD17 | PT1AD16 | PT1AD15 | PT1AD14 | PT1AD13 | PT1AD12 | PT1AD11 | PT1AD10 |
| 0x027A DDR0AD1 | R<br>W | DDR0AD17 | DDR0AD16 | DDR0AD15 | DDR0AD14 | DDR0AD13 | DDR0AD12 | DDR0AD11 | DDR0AD10 |
| 0x027B DDR1AD1 | R<br>W | DDR1AD17 | DDR1AD16 | DDR1AD15 | DDR1AD14 | DDR1AD13 | DDR1AD12 | DDR1AD11 | DDR1AD10 |
| 0x027C RDR0AD1 | R<br>W | RDR0AD17 | RDR0AD16 | RDR0AD15 | RDR0AD14 | RDR0AD13 | RDR0AD12 | RDR0AD11 | RDR0AD10 |
| 0x027D RDR1AD1 | R<br>W | RDR1AD17 | RDR1AD16 | RDR1AD15 | RDR1AD14 | RDR1AD13 | RDR1AD12 | RDR1AD11 | RDR1AD10 |
| 0x027E PER0AD1 | R<br>W | PER0AD17 | PER0AD16 | PER0AD15 | PER0AD14 | PER0AD13 | PER0AD12 | PER0AD1‘ | PER0AD10 |
| 0x027F PER1AD1 | R<br>W | PER1AD17 | PER1AD16 | PER1AD15 | PER1AD14 | PER1AD13 | PER1AD12 | PER1AD11 | PER1AD10 |
| 0x0280–0x0267 Non-PIM Address Range | R<br>W | Non-PIM Address Range | | | | | | | |

= Unimplemented or Reserved

| Register Name | | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|---|
| 0x0368 PTR | R | PTR7 | PTR6 | PTR5 | PTR4 | PTR3 | PTR2 | PTR1 | PTR0 |
| | W | | | | | | | | |
| 0x0369 PTIR | R | PTIR7 | PTIR6 | PTIR5 | PTIR4 | PTIR3 | PTIR2 | PTIR1 | PTIR0 |
| | W | | | | | | | | |
| 0x036A DDRR | R | DDRR7 | DDRR6 | DDRR5 | DDRR4 | DDRR3 | DDRR2 | DDRR1 | DDRR0 |
| | W | | | | | | | | |
| 0x036B RDRR | R | RDRR7 | RDRR6 | RDRR5 | RDRR4 | RDRR3 | RDRR2 | RDRR1 | RDRR0 |
| | W | | | | | | | | |
| 0x036C PERR | R | PERR7 | PERR6 | PERR5 | PERR4 | PERR3 | PERR2 | PERR1 | PERR0 |
| | W | | | | | | | | |
| 0x036D PPSR | R | PPSR7 | PPSR6 | PPSR5 | PPSR4 | PPSR3 | PPSR2 | PPSR1 | PPSR0 |
| | W | | | | | | | | |
| 0x036E Reserved | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | W | | | | | | | | |
| 0x036F PTRRR | R | PTRRR7 | PTRRR6 | PTRRR5 | PTRRR4 | PTRRR3 | PTRRR2 | PTRRR1 | PTRRR0 |
| | W | | | | | | | | |
| 0x0370 PTL | R | PTL7 | PTL6 | PTL5 | PTL4 | PTL3 | PTL2 | PTL1 | PTL0 |
| | W | | | | | | | | |
| 0x0371 PTIL | R | PTIL7 | PTIL6 | PTIL5 | PTIL4 | PTIL3 | PTIL2 | PTIL1 | PTIL0 |
| | W | | | | | | | | |
| 0x0372 DDRL | R | DDRL7 | DDRL6 | DDRL5 | DDRL4 | DDRL3 | DDRL2 | DDRL1 | DDRL0 |
| | W | | | | | | | | |
| 0x0373 RDRL | R | RDRL7 | RDRL6 | RDRL5 | RDRL4 | RDRL3 | RDRL2 | RDRL1 | RDRL0 |
| | W | | | | | | | | |
| 0x0374 PERL | R | PERL7 | PERL6 | PERL5 | PERL4 | PERL3 | PERL2 | PERL1 | PERL0 |
| | W | | | | | | | | |
| 0x0375 PPSL | R | PPSL7 | PPSL6 | PPSL5 | PPSL4 | PPSL3 | PPSL2 | PPSL1 | PPSL0 |
| | W | | | | | | | | |
| 0x0376 WOML | R | WOML7 | WOML6 | WOML5 | WOML4 | WOML3 | WOML2 | WOML1 | WOML0 |
| | W | | | | | | | | |
| 0x0377 PTLRR | R | PTLRR7 | PTLRR6 | PTLRR5 | PTLRR4 | 0 | 0 | 0 | 0 |
| | W | | | | | | | | |

= Unimplemented or Reserved

| Register Name | | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|---|
| 0x0378 PTF | R | PTF7 | PTF6 | PTF5 | PTF4 | PTF3 | PTF2 | PTF1 | PTF0 |
| | W | | | | | | | | |
| 0x0379 PTIF | R | PTIF7 | PTIF6 | PTIF5 | PTIF4 | PTIF3 | PTIF2 | PTIF1 | PTIF0 |
| | W | | | | | | | | |
| 0x037A DDRF | R | DDRF7 | DDRF6 | DDRF5 | DDRF4 | DDRF3 | DDRF2 | DDRF1 | DDRF0 |
| | W | | | | | | | | |
| 0x037B RDRF | R | RDRF7 | RDRF6 | RDRF5 | RDRF4 | RDRF3 | RDRF2 | RDRF1 | RDRF0 |
| | W | | | | | | | | |
| 0x037C PERF | R | PERF7 | PERF6 | PERF5 | PERF4 | PERF3 | PERF2 | PERF1 | PERF0 |
| | W | | | | | | | | |
| 0x037D PPSF | R | PPSF7 | PPSF6 | PPSF5 | PPSF4 | PPSF3 | PPSF2 | PPSF1 | PPSF0 |
| | W | | | | | | | | |
| 0x037E Reserved | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | W | | | | | | | | |
| 0x037F PTFRR | R | 0 | 0 | PTFRR5 | PTFRR4 | PTFRR3 | PTFRR2 | PTFRR1 | PTFRR0 |
| | W | | | | | | | | |

= Unimplemented or Reserved

## 2.3.2 Register Descriptions

The following table summarizes the effect of the various configuration bits, i.e. data direction (DDR), output level (IO), reduced drive (RDR), pull enable (PE), pull select (PS) on the pin function and pull device activity.

The configuration bit PS is used for two purposes:

1. Configure the sensitive interrupt edge (rising or falling), if interrupt is enabled.
2. Select either a pull-up or pull-down device if PE is active.

**Table 2-3. Pin Configuration Summary**

| DDR | IO | RDR | PE | PS[1] | IE[2] | Function | Pull Device | Interrupt |
|---|---|---|---|---|---|---|---|---|
| 0 | x | x | 0 | x | 0 | Input | Disabled | Disabled |
| 0 | x | x | 1 | 0 | 0 | Input | Pull Up | Disabled |
| 0 | x | x | 1 | 1 | 0 | Input | Pull Down | Disabled |
| 0 | x | x | 0 | 0 | 1 | Input | Disabled | Falling edge |
| 0 | x | x | 0 | 1 | 1 | Input | Disabled | Rising edge |
| 0 | x | x | 1 | 0 | 1 | Input | Pull Up | Falling edge |
| 0 | x | x | 1 | 1 | 1 | Input | Pull Down | Rising edge |
| 1 | 0 | 0 | x | x | 0 | Output, full drive to 0 | Disabled | Disabled |
| 1 | 1 | 0 | x | x | 0 | Output, full drive to 1 | Disabled | Disabled |
| 1 | 0 | 1 | x | x | 0 | Output, reduced drive to 0 | Disabled | Disabled |
| 1 | 1 | 1 | x | x | 0 | Output, reduced drive to 1 | Disabled | Disabled |
| 1 | 0 | 0 | x | 0 | 1 | Output, full drive to 0 | Disabled | Falling edge |
| 1 | 1 | 0 | x | 1 | 1 | Output, full drive to 1 | Disabled | Rising edge |
| 1 | 0 | 1 | x | 0 | 1 | Output, reduced drive to 0 | Disabled | Falling edge |
| 1 | 1 | 1 | x | 1 | 1 | Output, reduced drive to 1 | Disabled | Rising edge |

1. Always “0” on Port A, B, C, D, E, K, AD0, and AD1.
2. Applicable only on Port P, H, and J.

**NOTE**

All register bits in this module are completely synchronous to internal clocks during a register read.

## 2.3.3 Port A Data Register (PORTA)

Address 0x0000 (PRR) Access: User read/write(1)

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | PA7 | PA6 | PA5 | PA4 | PA3 | PA2 | PA1 | PA0 |
| Altern. Function | ADDR15 mux IVD15 | ADDR14 mux IVD14 | ADDR13 mux IVD13 | ADDR12 mux IVD12 | ADDR11 mux IVD11 | ADDR10 mux IVD10 | ADDR9 mux IVD9 | ADDR8 mux IVD8 |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 2-1. Port A Data Register (PORTA)**

1. Read: Anytime. In emulation modes, read operations will return the data from the external bus, in all other modes the data source is depending on the data direction value.
   Write: Anytime. In emulation modes, write operations will also be directed to the external bus.

**Table 2-4. PORTA Register Field Descriptions**

| Field | Description |
|---|---|
| 7-0<br>PA | **Port A general purpose input/output data**—Data Register<br>Port A pins 7 through 0 are associated with address outputs ADDR[15:8] respectively in expanded modes. In emulation modes the address is multiplexed with IVD[15:8].<br>When not used with the alternative function, these pins can be used as general purpose I/O.<br>If the associated data direction bits of these pins are set to 1, a read returns the value of the port register, otherwise the buffered pin input state is read. |

## 2.3.4 Port B Data Register (PORTB)

Address 0x0001 (PRR) Access: User read/write(1)

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | PB7 | PB6 | PB5 | PB4 | PB3 | PB2 | PB1 | PB0 |
| Altern. Function | ADDR7 mux IVD7 | ADDR6 mux IVD6 | ADDR5 mux IVD5 | ADDR4 mux IVD4 | ADDR3 mux IVD3 | ADDR2 mux IVD2 | ADDR1 mux IVD1 | ADDR0 mux IVD0 or $\overline{\text{UDS}}$ |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 2-2. Port B Data Register (PORTB)**

1. Read: Anytime. In emulation modes, read operations will return the data from the external bus, in all other modes the data source is depending on the data direction value.
   Write: Anytime. In emulation modes, write operations will also be directed to the external bus.

**Table 2-5. PORTB Register Field Descriptions**

| Field | Description |
|---|---|
| 7-0<br>PB | **Port B general purpose input/output data**—Data Register<br>Port B pins 7 through 0 are associated with address outputs ADDR[7:0] respectively in expanded modes. In emulation modes the address is multiplexed with IVD[7:0]. In normal expanded mode pin 0 is related to the $\overline{UDS}$ input.<br>When not used with the alternative function, these pins can be used as general purpose I/O.<br>If the associated data direction bits of these pins are set to 1, a read returns the value of the port register, otherwise the buffered pin input state is read. |

## 2.3.5 Port A Data Direction Register (DDRA)

Address 0x0002 (PRR) Access: User read/write(1)

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | DDRA7 | DDRA6 | DDRA5 | DDRA4 | DDRA3 | DDRA2 | DDRA1 | DDRA0 |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 2-3. Port A Data Direction Register (DDRA)**

1. Read: Anytime. In emulation modes, read operations will return the data from the external bus, in all other modes the data source is depending on the data direction value.
   Write: Anytime. In emulation modes, write operations will also be directed to the external bus.

**Table 2-6. DDRA Register Field Descriptions**

| Field | Description |
|---|---|
| 7-0<br>DDRA | **Port A Data Direction**—<br>This register controls the data direction of pins 7 through 0.<br>The external bus function forces the I/O state to be outputs for all associated pins. In this case the data direction bits will not change.<br>When operating a pin as a general purpose I/O, the associated data direction bit determines whether it is an input or output.<br>1 Associated pin is configured as output.<br>0 Associated pin is configured as high-impedance input. |

## 2.3.6 Port B Data Direction Register (DDRB)

Address 0x0003 (PRR) Access: User read/write(1)

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | DDRB7 | DDRB6 | DDRB5 | DDRB4 | DDRB3 | DDRB2 | DDRB1 | DDRB0 |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 2-4. Port B Data Direction Register (DDRB)**

1. Read: Anytime. In emulation modes, read operations will return the data from the external bus, in all other modes the data source is depending on the data direction value.
   Write: Anytime. In emulation modes, write operations will also be directed to the external bus.

**Table 2-7. DDRB Register Field Descriptions**

| Field | Description |
|---|---|
| 7-0<br>DDRB | **Port B Data Direction**—<br>This register controls the data direction of pins 7 through 0.<br>The external bus function forces the I/O state to be outputs for all associated pins. In this case the data direction bits will not change.<br>When operating a pin as a general purpose I/O, the associated data direction bit determines whether it is an input or output.<br>1 Associated pin is configured as output.<br>0 Associated pin is configured as high-impedance input. |

## 2.3.7 Port C Data Register (PORTC)

Address 0x0004 (PRR) Access: User read/write(1)

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | PC7 | PC6 | PC5 | PC4 | PC3 | PC2 | PC1 | PC0 |
| Altern. Function | DATA15 | DATA14 | DATA13 | DATA12 | DATA11 | DATA10 | DATA9 | DATA8 |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 2-5. Port C Data Register (PORTC)**

1. Read: Anytime. In emulation modes, read operations will return the data from the external bus, in all other modes the data source is depending on the data direction value.
   Write: Anytime. In emulation modes, write operations will also be directed to the external bus.

**Table 2-8. PORTC Register Field Descriptions**

| Field | Description |
|---|---|
| 7-0<br>PC | **Port C general purpose input/output data**—Data Register<br>Port C pins 7 through 0 are associated with data I/O lines DATA[15:8] respectively in expanded modes.<br>When not used with the alternative function, these pins can be used as general purpose I/O.<br>If the associated data direction bits of these pins are set to 1, a read returns the value of the port register, otherwise the buffered pin input state is read. |

## 2.3.8 Port D Data Register (PORTD)

Address 0x0005 (PRR) Access: User read/write(1)

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | PD7 | PD6 | PD5 | PD4 | PD3 | PD2 | PD1 | PD0 |
| Altern. Function | DATA7 | DATA6 | DATA5 | DATA4 | DATA3 | DATA2 | DATA1 | DATA0 |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 2-6. Port D Data Register (PORTD)**

1. Read: Anytime. In emulation modes, read operations will return the data from the external bus, in all other modes the data source is depending on the data direction value.
   Write: Anytime. In emulation modes, write operations will also be directed to the external bus.

**Table 2-9. PORTD Register Field Descriptions**

| Field | Description |
|---|---|
| 7-0<br>PD | **Port D general purpose input/output data**—Data Register<br>Port D pins 7 through 0 are associated with data I/O lines DATA[7:0] respectively in expanded modes.<br>When not used with the alternative function, these pins can be used as general purpose I/O.<br>If the associated data direction bits of these pins are set to 1, a read returns the value of the port register, otherwise the buffered pin input state is read. |

## 2.3.9 Port C Data Direction Register (DDRC)

Address 0x0006 (PRR) Access: User read/write(1)

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | DDRC7 | DDRC6 | DDRC5 | DDRC4 | DDRC3 | DDRC2 | DDRC1 | DDRC0 |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 2-7. Port C Data Direction Register (DDRC)**

1. Read: Anytime. In emulation modes, read operations will return the data from the external bus, in all other modes the data source is depending on the data direction value.
   Write: Anytime. In emulation modes, write operations will also be directed to the external bus.

**Table 2-10. DDRC Register Field Descriptions**

| Field | Description |
|---|---|
| 7-0<br>DDRC | Port C Data Direction—<br>This register controls the data direction of pins 7 through 0.<br>The external bus function controls the data direction for the associated pins. In this case the data direction bits will not change.<br>When operating a pin as a general purpose I/O, the associated data direction bit determines whether it is an input or output.<br>1 Associated pin is configured as output.<br>0 Associated pin is configured as high-impedance input. |

## 2.3.10 Port D Data Direction Register (DDRD)

Address 0x0007 (PRR) Access: User read/write(1)

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | DDRD7 | DDRD6 | DDRD5 | DDRD4 | DDRD3 | DDRD2 | DDRD1 | DDRD0 |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 2-8. Port D Data Direction Register (DDRD)**

1. Read: Anytime. In emulation modes, read operations will return the data from the external bus, in all other modes the data source is depending on the data direction value.
   Write: Anytime. In emulation modes, write operations will also be directed to the external bus.

**Table 2-11. DDRD Register Field Descriptions**

| Field | Description |
|---|---|
| 7-0<br>DDRD | **Port D Data Direction**—<br>This register controls the data direction of pins 7 through 0.<br>When used with the external bus this function controls the data direction for the associated pins. In this case the data direction bits will not change.<br>When operating a pin as a general purpose I/O, the associated data direction bit determines whether it is an input or output.<br>1 Associated pin is configured as output.<br>0 Associated pin is configured as high-impedance input. |

## 2.3.11 Port E Data Register (PORTE)

Address 0x0008 (PRR) Access: User read/write(1)

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | PE7 | PE6 | PE5 | PE4 | PE3 | PE2 | PE1 | PE0 |
| W | | | | | | | | |
| Altern. Function | $\overline{XCLKS}$ or ECLKX2 | MODB or $\overline{TAGHI}$ | MODA or $\overline{RE}$ or $\overline{TAGLO}$ | ECLK | EROMCTL or $\overline{LSTRB}$ or $\overline{LDS}$ | R$\overline{W}$ or $\overline{WE}$ | $\overline{IRQ}$ | $\overline{XIRQ}$ |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | —(2) | —2 |

= Unimplemented or Reserved

**Figure 2-9. Port E Data Register (PORTE)**

1. Read: Anytime. In emulation modes, read operations will return the data from the external bus, in all other modes the data source is depending on the data direction value.
   Write: Anytime. In emulation modes, write operations will also be directed to the external bus.
2. These registers are reset to zero. Two bus clock cycles after reset release the register values are updated with the associated pin values.

**Table 2-12. PORTE Register Field Descriptions**

| Field | Description |
|---|---|
| 7-2<br>PE | **Port E general purpose input/output data**—Data Register<br>Port E bits 7 through 0 are associated with external bus control signals and interrupt inputs. These include mode select (MODB, MODA), E clock, double frequency E clock, Instruction Tagging High and Low ($\overline{TAGHI}$, $\overline{TAGLO}$), Read/Write (R$\overline{W}$), Read Enable and Write Enable ($\overline{RE}$, $\overline{WE}$), Lower Data Select ($\overline{LDS}$).<br>When not used with the alternative functions, Port E pins 7-2 can be used as general purpose I/O.<br>If the associated data direction bits of these pins are set to 1, a read returns the value of the port register, otherwise the buffered pin input state is read.<br>Pins 6 and 5 are inputs with enabled pull-down devices while RESET pin is low.<br>Pins 7 and 3 are inputs with enabled pull-up devices while RESET pin is low. |
| 1<br>PE | **Port E general purpose input data and interrupt**—Data Register, $\overline{IRQ}$ input.<br>This pin can be used as general purpose and $\overline{IRQ}$ input. |
| 0<br>PE | **Port E general purpose input data and interrupt**—Data Register, $\overline{XIRQ}$ input.<br>This pin can be used as general purpose and $\overline{XIRQ}$ input. |

## 2.3.12 Port E Data Direction Register (DDRE)

Address 0x0009 (PRR) Access: User read/write(1)

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | DDRE7 | DDRE6 | DDRE5 | DDRE4 | DDRE3 | DDRE2 | 0 | 0 |
| W | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented or Reserved

**Figure 2-10. Port E Data Direction Register (DDRE)**

1. Read: Anytime. In emulation modes, read operations will return the data from the external bus, in all other modes the data source is depending on the data direction value.
   Write: Anytime. In emulation modes, write operations will also be directed to the external bus.

**Table 2-13. DDRE Register Field Descriptions**

| Field | Description |
|---|---|
| 7-2 DDRE | **Port E Data Direction**—<br>This register controls the data direction of pins 7 through 2.<br>The external bus function controls the data direction for the associated pins. In this case the data direction bits will not change.<br>When operating a pin as a general purpose I/O, the associated data direction bit determines whether it is an input or output.<br>1 Associated pin is configured as output.<br>0 Associated pin is configured as high-impedance input. |
| 1-0 | **Reserved**—<br>Port E bit 1 (associated with $\overline{IRQ}$) and bit 0 (associated with $\overline{XIRQ}$) cannot be configured as outputs. Port E, bits 1 and 0, can be read regardless of whether the alternate interrupt function is enabled. |

## 2.3.13 S12X_EBI ports, BKGD pin Pull-up Control Register (PUCR)

Address 0x000C (PRR) Access: User read/write(1)

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | PUPKE | BKPUE | 0 | PUPEE | PUPDE | PUPCE | PUPBE | PUPAE |
| W | | | | | | | | |
| Reset | 1 | 1 | 0 | 1 | 0 | 0 | 0 | 0 |

= Unimplemented or Reserved

**Figure 2-11. S12X_EBI ports, BKGD pin Pull-up Control Register (PUCR)**

1. Read:Anytime in single-chip modes.
   Write:Anytime, except BKPUE which is writable in Special Test Mode only.

**Table 2-14. PUCR Register Field Descriptions**

| Field | Description |
|---|---|
| 7<br>PUPKE | **Pull-up Port K Enable**—Enable pull-up devices on all Port K input pins<br>This bit configures whether pull-up devices are activated, if the pins are used as inputs. This bit has no effect if the pins are used as outputs. Out of reset the pull-up devices are enabled.<br>1 Pull-up devices enabled.<br>0 Pull-up devices disabled. |
| 6<br>BKPUE | **BKGD pin pull-up Enable**—Enable pull-up devices on BKGD pin<br>This bit configures whether a pull-up device is activated, if the pin is used as input. This bit has no effect if the pin is used as outputs. Out of reset the pull-up device is enabled.<br>1 Pull-up device enabled.<br>0 Pull-up device disabled. |
| 5 | **Reserved**— |
| 4<br>PUPEE | **Pull-up Port E Enable**—Enable pull-up devices on all Port E input pins except on pins 5 and 6 which have pull-down devices only enabled during reset. This bit has no effect on these pins.<br>This bit configures whether pull-up devices are activated, if the pins are used as inputs. This bit has no effect if the pins are used as outputs. Out of reset the pull-up devices are enabled.<br>1 Pull-up devices enabled.<br>0 Pull-up devices disabled. |
| 3<br>PUPDE | **Pull-up Port D Enable**—Enable pull-up devices on all Port D input pins<br>This bit configures whether pull-up devices are activated, if the pins are used as inputs. This bit has no effect if the pins are used as outputs. Out of reset the pull-up devices are disabled.<br>1 Pull-up devices enabled.<br>0 Pull-up devices disabled. |
| 2<br>PUPCE | **Pull-up Port C Enable**—Enable pull-up devices on all Port C input pins<br>This bit configures whether pull-up devices are activated, if the pins are used as inputs. This bit has no effect if the pins are used as outputs. Out of reset the pull-up devices are disabled.<br>1 Pull-up devices enabled.<br>0 Pull-up devices disabled. |
| 1<br>PUPBE | **Pull-up Port B Enable**—Enable pull-up devices on all Port B input pins<br>This bit configures whether pull-up devices are activated, if the pins are used as inputs. This bit has no effect if the pins are used as outputs. Out of reset the pull-up devices are disabled.<br>1 Pull-up devices enabled.<br>0 Pull-up devices disabled. |
| 0<br>PUPAE | **Pull-up Port A Enable**—Enable pull-up devices on all Port A input pins<br>This bit configures whether pull-up devices are activated, if the pins are used as inputs. This bit has no effect if the pins are used as outputs. Out of reset the pull-up devices are disabled.<br>1 Pull-up devices enabled.<br>0 Pull-up devices disabled. |

## 2.3.14 S12X_EBI ports Reduced Drive Register (RDRIV)

Address 0x000D (PRR) Access: User read/write(1)

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | RDPK | 0 | 0 | RDPE | RDPD | RDPC | RDPB | RDPA |
| W | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented or Reserved

**Figure 2-12. S12X_EBI ports Reduced Drive Register (RDRIV)**

1. Read: Anytime. In emulation modes, read operations will return the data from the external bus, in all other modes the data source is depending on the data direction value.
   Write: Anytime. In emulation modes, write operations will also be directed to the external bus.

This register is used to select reduced drive for the pins associated with the S12X_EBI ports A, B, C, D, E, and K. If enabled, the pins drive at approx. 1/5 of the full drive strength.

The reduced drive functionality does not take effect on the pins in emulation modes.

**Table 2-15. RDRIV Register Field Descriptions**

| Field | Description |
|---|---|
| 7<br>RDPK | **Port K reduced drive**—Select reduced drive for outputs<br>This bit configures the drive strength of all Port K output pins as either full or reduced independent of the function used on the pins. If a pin is used as input this bit has no effect.<br>1 Reduced drive selected (approx. 1/5 of the full drive strength).<br>0 Full drive strength enabled. |
| 6-5 | **Reserved**— |
| 4<br>RDPE | **Port E reduced drive**—Select reduced drive for outputs<br>This bit configures the drive strength of all Port E output pins as either full or reduced independent of the function used on the pins. If a pin is used as input this bit has no effect.<br>1 Reduced drive selected (approx. 1/5 of the full drive strength).<br>0 Full drive strength enabled. |
| 3<br>RDPD | **Port D reduced drive**—Select reduced drive for outputs<br>This bit configures the drive strength of all output pins as either full or reduced independent of the function used on the pins. If a pin is used as input this bit has no effect.<br>1 Reduced drive selected (approx. 1/5 of the full drive strength).<br>0 Full drive strength enabled. |
| 2<br>RDPC | **Port C reduced drive**—Select reduced drive for outputs<br>This bit configures the drive strength of all output pins as either full or reduced independent of the function used on the pins. If a pin is used as input this bit has no effect.<br>1 Reduced drive selected (approx. 1/5 of the full drive strength).<br>0 Full drive strength enabled. |

**Table 2-15. RDRIV Register Field Descriptions (continued)**

| Field | Description |
|---|---|
| 1<br>RDPB | **Port B reduced drive**—Select reduced drive for outputs<br>This bit configures the drive strength of all output pins as either full or reduced independent of the function used on the pins. If a pin is used as input this bit has no effect.<br>1 Reduced drive selected (approx. 1/5 of the full drive strength).<br>0 Full drive strength enabled. |
| 0<br>RDPA | **Port A reduced drive**—Select reduced drive for outputs<br>This bit configures the drive strength of all output pins as either full or reduced independent of the function used on the pins. If a pin is used as input this bit has no effect.<br>1 Reduced drive selected (approx. 1/5 of the full drive strength).<br>0 Full drive strength enabled. |

## 2.3.15 ECLK Control Register (ECLKCTL)

Address 0x001C (PRR) Access: User read/write(1)

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | NECLK | NCLKX2 | DIV16 | EDIV4 | EDIV3 | EDIV2 | EDIV1 | EDIV0 |
| Reset(2): | Mode Depen-dent | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| SS | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| ES | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| ST | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| EX | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| NS | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| NX | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented or Reserved

**Figure 2-13. ECLK Control Register (ECLKCTL)**

1. Read: Anytime. In emulation modes, read operations will return the data from the external bus, in all other modes the data source is depending on the data direction value.
   Write: Anytime. In emulation modes, write operations will also be directed to the external bus.
2. Reset values in emulation modes are identical to those of the target mode.

The ECLKCTL register is used to control the availability of the free-running clocks and the free-running clock divider.

**Table 2-16. ECLKCTL Register Field Descriptions**

| Field | Description |
|---|---|
| 7<br>NECLK | **No ECLK**—Disable ECLK output<br>This bit controls the availability of a free-running clock on the ECLK pin.<br>Clock output is always active in emulation modes and if enabled in all other operating modes.<br>1 ECLK disabled<br>0 ECLK enabled |
| 6<br>NCLKX2 | **No ECLKX2**—Disable ECLKX2 output<br>This bit controls the availability of a free-running clock on the ECLKX2 pin. This clock has a fixed rate of twice the internal Bus Clock.<br>Clock output is always active in emulation modes and if enabled in all other operating modes.<br>1 ECLKX2 disabled<br>0 ECLKX2 enabled |
| 5<br>DIV16 | **Free-running ECLK predivider**—Divide by 16<br>This bit enables a divide-by-16 stage on the selected EDIV rate.<br>1 Divider enabled: ECLK rate = EDIV rate divided by 16<br>0 Divider disabled: ECLK rate = EDIV rate |
| 4-0<br>EDIV | **Free-running ECLK Divider**—Configure ECLK rate<br>These bits determine the rate of the free-running clock on the ECLK pin. Divider is always disabled in emulation modes and active as programmed in all other operating modes.<br>00000 ECLK rate = Bus Clock rate<br>00001 ECLK rate = Bus Clock rate divided by 2<br>00010 ECLK rate = Bus Clock rate divided by 3, ...<br>11111 ECLK rate = Bus Clock rate divided by 32 |

## 2.3.16 PIM Reserved Register

Address 0x001D (PRR) Access: User read(1)

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| W | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented or Reserved

**Figure 2-14. PIM Reserved Register**

1. Read: Always reads 0x00
   Write: Unimplemented

## 2.3.17 IRQ Control Register (IRQCR)

Address 0x001E — Access: User read/write(1)

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | IRQE | IRQEN | 0 | 0 | 0 | 0 | 0 | 0 |
| W | | | | | | | | |
| Reset | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented or Reserved

**Figure 2-15. IRQ Control Register (IRQCR)**

1. Read: See individual bit descriptions below.
   Write: See individual bit descriptions below.

**Table 2-17. IRQCR Register Field Descriptions**

| Field | Description |
|---|---|
| 7<br>IRQE | **IRQ select edge sensitive only**—<br>Special modes: Read or write anytime.<br>Normal & emulation modes: Read anytime, write once.<br>1 $\overline{IRQ}$ configured to respond only to falling edges. Falling edges on the $\overline{IRQ}$ pin will be detected anytime IRQE = 1 and will be cleared only upon a reset or the servicing of the $\overline{IRQ}$ interrupt.<br>0 $\overline{IRQ}$ configured for low level recognition. |
| 6<br>IRQEN | **External IRQ enable**—<br>Read or write anytime.<br>1 External IRQ pin is connected to interrupt logic.<br>0 External IRQ pin is disconnected from interrupt logic. |
| 5-0 | **Reserved**— |

## 2.3.18 PIM Reserved Register

This register is reserved for factory testing of the PIM module and is not available in normal operation.

Address 0x001F — Access: User read(1)

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| W | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented or Reserved

**Figure 2-16. PIM Reserved Register**

1. Read: Always reads 0x00
   Write: Unimplemented

**NOTE**

Writing to this register when in special modes can alter the pin functionality.

## 2.3.19 Port K Data Register (PORTK)

Address 0x0032 (PRR) Access: User read/write(1)

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | PK7 | PK6 | PK5 | PK4 | PK3 | PK2 | PK1 | PK0 |
| Altern. Function | ROMCTL or $\overline{\text{EWAIT}}$ | ADDR22 mux ACC2 | ADDR21 mux ACC1 | ADDR20 mux ACC0 | ADDR19 mux IQSTAT3 | ADDR18 mux IQSTAT2 | ADDR17 mux IQSTAT1 | ADDR16 mux IQSTAT0 |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 2-17. Port K Data Register (PORTK)**

1. Read: Anytime. In emulation modes, read operations will return the data from the external bus, in all other modes the data source is depending on the data direction value.
   Write: Anytime. In emulation modes, write operations will also be directed to the external bus.

**Table 2-18. PORTK Register Field Descriptions**

| Field | Description |
|---|---|
| 7-0<br>PK | **Port K general purpose input/output data**—Data Register<br>Port K pins 7 through 0 are associated with external bus control signals and internal memory expansion emulation pins. These include ADDR[22:16], Access Source (ACC[2:0]), External Wait ($\overline{\text{EWAIT}}$) and instruction pipe signals IQSTAT[3:0]. Bits 6-0 carry the external addresses in all expanded modes. In emulation modes the address is multiplexed with the alternate functions ACC and IQSTAT on the respective pins.<br>When not used with the alternative function, these pins can be used as general purpose I/O.<br>If the associated data direction bits of these pins are set to 1, a read returns the value of the port register, otherwise the buffered pin input state is read. |

## 2.3.20 Port K Data Direction Register (DDRK)

Address 0x0033 (PRR) Access: User read/write(1)

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | DDRK7 | DDRK6 | DDRK5 | DDRK4 | DDRK3 | DDRK2 | DDRK1 | DDRK0 |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 2-18. Port K Data Direction Register (DDRK)**

1. Read: Anytime. In emulation modes, read operations will return the data from the external bus, in all other modes the data source is depending on the data direction value.
   Write: Anytime. In emulation modes, write operations will also be directed to the external bus.

**Table 2-19. DDRK Register Field Descriptions**

| Field | Description |
|---|---|
| 7-0<br>DDRK | **Port K Data Direction**—<br>This register controls the data direction of pins 7 through 0.<br>The external bus function controls the data direction for the associated pins. In this case the data direction bits will not change.<br>When operating a pin as a general purpose I/O, the associated data direction bit determines whether it is an input or output.<br>1 Associated pin is configured as output.<br>0 Associated pin is configured as high-impedance input. |

## 2.3.21 Port T Data Register (PTT)

Address 0x0240 Access: User read/write(1)

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | PTT7 | PTT6 | PTT5 | PTT4 | PTT3 | PTT2 | PTT1 | PTT0 |
| Altern. Function | IOC7 | IOC6 | IOC5 | IOC4 | IOC3 | IOC2 | IOC1 | IOC0 |
| | — | — | VREG_API | — | — | — | — | — |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 2-19. Port T Data Register (PTT)**

1. Read: Anytime.
   Write: Anytime.

**Table 2-20. PTT Register Field Descriptions**

| Field | Description |
|---|---|
| 7-6<br>PTT | **Port T general purpose input/output data**—Data Register<br>Port T pins 7 through 0 are associated with ECT channels IOC7 and IOC6.<br>When not used with the alternative function, these pins can be used as general purpose I/O.<br>If the associated data direction bit of this pin is set to 1, a read returns the value of the port register, otherwise the buffered pin input state is read. |
| 5<br>PTT | **Port T general purpose input/output data**—Data Register<br>Port T pins 5 is associated with ECT channel IOC5 and the VREG_API output.<br>The ECT function takes precedence over the VREG_API and the general purpose I/O function if the related channel is enabled.<br>When not used with the alternative function, these pins can be used as general purpose I/O.<br>If the associated data direction bit of this pin is set to 1, a read returns the value of the port register, otherwise the buffered pin input state is read. |
| 4-0<br>PTT | **Port T general purpose input/output data**—Data Register<br>Port T pins 4 through 0 are associated with ECT channels IOC4 through IOC0.<br>When not used with the alternative function, these pins can be used as general purpose I/O.<br>If the associated data direction bit of this pin is set to 1, a read returns the value of the port register, otherwise the buffered pin input state is read. |

## 2.3.22 Port T Input Register (PTIT)

Address 0x0241 Access: User read(1)

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | PTIT7 | PTIT6 | PTIT5 | PTIT4 | PTIT3 | PTIT2 | PTIT1 | PTIT0 |
| W | | | | | | | | |
| Reset | u | u | u | u | u | u | u | u |

= Unimplemented or Reserved u = Unaffected by reset

**Figure 2-20. Port T Input Register (PTIT)**

1. Read: Anytime.
   Write:Never, writes to this register have no effect.

**Table 2-21. PTIT Register Field Descriptions**

| Field | Description |
|---|---|
| 7-0<br>PTIT | **Port T input data**—<br>This register always reads back the buffered state of the associated pins. This can also be used to detect overload or short circuit conditions on output pins. |

## 2.3.23 Port T Data Direction Register (DDRT)

Address 0x0242 Access: User read/write(1)

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | DDRT7 | DDRT6 | DDRT5 | DDRT4 | DDRT3 | DDRT2 | DDRT1 | DDRT0 |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 2-21. Port T Data Direction Register (DDRT)**

1. Read: Anytime.
   Write: Anytime.

**Table 2-22. DDRT Register Field Descriptions**

| Field | Description |
|---|---|
| 7-0<br>DDRT | **Port T data direction**—<br>This register controls the data direction of pins 7 through 0.<br>The ECT forces the I/O state to be an output for each timer port associated with an enabled output compare. In this case the data direction bits will not change.<br>The data direction bits revert to controlling the I/O direction of a pin when the associated timer output compare is disabled.<br>The timer Input Capture always monitors the state of the pin.<br>1 Associated pin is configured as output.<br>0 Associated pin is configured as high-impedance input. |

**NOTE**

Due to internal synchronization circuits, it can take up to 2 bus clock cycles until the correct value is read on PTT or PTIT registers, when changing the DDRT register.

## 2.3.24 Port T Reduced Drive Register (RDRT)

Address 0x0243 Access: User read/write(1)

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | RDRT7 | RDRT6 | RDRT5 | RDRT4 | RDRT3 | RDRT2 | RDRT1 | RDRT0 |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 2-22. Port T Reduced Drive Register (RDRT)**

1. Read: Anytime.
   Write: Anytime.

**Table 2-23. RDRT Register Field Descriptions**

| Field | Description |
|---|---|
| 7-0<br>RDRT | **Port T reduced drive**—Select reduced drive for outputs<br>This register configures the drive strength of output pins 7 through 0 as either full or reduced independent of the function used on the pins. If a pin is used as input this bit has no effect.<br>1 Reduced drive selected (approx. 1/5 of the full drive strength).<br>0 Full drive strength enabled. |

## 2.3.25 Port T Pull Device Enable Register (PERT)

Address 0x0244 Access: User read/write(1)

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | PERT7 | PERT6 | PERT5 | PERT4 | PERT3 | PERT2 | PERT1 | PERT0 |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 2-23. Port T Pull Device Enable Register (PERT)**

1. Read: Anytime.
   Write: Anytime.

**Table 2-24. PERT Register Field Descriptions**

| Field | Description |
|---|---|
| 7-0<br>PERT | **Port T pull device enable**—Enable pull devices on input pins<br>These bits configure whether a pull device is activated, if the associated pin is used as an input. This bit has no effect if the pin is used as an output. Out of reset no pull device is enabled.<br>1 Pull device enabled.<br>0 Pull device disabled. |

## 2.3.26 Port T Polarity Select Register (PPST)

Address 0x0245 Access: User read/write(1)

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | PPST7 | PPST6 | PPST5 | PPST4 | PPST3 | PPST2 | PPST1 | PPST0 |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 2-24. Port T Polarity Select Register (PPST)**

1. Read: Anytime.
   Write: Anytime.

**Table 2-25. PPST Register Field Descriptions**

| Field | Description |
|---|---|
| 7-0<br>PPST | **Port T pull device select**—Determine pull device polarity on input pins<br>This register selects whether a pull-down or a pull-up device is connected to the pin.<br>1 A pull-down device is connected to the associated pin, if enabled and if the pin is used as input.<br>0 A pull-up device is connected to the associated pin, if enabled and if the pin is used as input. |

## 2.3.27 PIM Reserved Register

Address 0x0246 Access: User read(1)

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| W | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented or Reserved

**Figure 2-25. PIM Reserved Register**

1. Read: Always reads 0x00
   Write: Unimplemented

## 2.3.28 PIM Reserved Register

Address 0x0247 Access: User read(1)

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| W | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented or Reserved

**Figure 2-26. PIM Reserved Register**

1. Read: Always reads 0x00
   Write: Unimplemented

## 2.3.29 Port S Data Register (PTS)

Address 0x0248 Access: User read/write(1)

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | PTS7 | PTST6 | PTS5 | PTS4 | PTS3 | PTS2 | PTS1 | PTS0 |
| Altern. Function | $\overline{SS0}$ | SCK0 | MOSI0 | MISO0 | TXD1 | RXD1 | TXD0 | RXD0 |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 2-27. Port S Data Register (PTS)**

1. Read: Anytime.
   Write: Anytime.

**Table 2-26. PTS Register Field Descriptions**

| Field | Description |
|---|---|
| 7<br>PTS | **Port S general purpose input/output data**—Data Register<br>Port S pin 7 is associated with the $\overline{SS}$ signal of the SPI0 module.<br>When not used with the alternative function, this pin can be used as general purpose I/O.<br>If the associated data direction bit of this pin is set to 1, a read returns the value of the port register, otherwise the buffered pin input state is read. |
| 6<br>PTS | **Port S general purpose input/output data**—Data Register<br>Port S pin 6 is associated with the SCK signal of the SPI0 module.<br>When not used with the alternative function, this pin can be used as general purpose I/O.<br>If the associated data direction bit of this pin is set to 1, a read returns the value of the port register, otherwise the buffered pin input state is read. |
| 5<br>PTS | **Port S general purpose input/output data**—Data Register<br>Port S pin 5 is associated with the MOSI signal of the SPI0 module.<br>When not used with the alternative function, this pin can be used as general purpose I/O.<br>If the associated data direction bit of this pin is set to 1, a read returns the value of the port register, otherwise the buffered pin input state is read. |
| 4<br>PTS | **Port S general purpose input/output data**—Data Register<br>Port S pin 4 is associated with the MISO signal of the SPI0 module.<br>When not used with the alternative function, this pin can be used as general purpose I/O.<br>If the associated data direction bit of this pin is set to 1, a read returns the value of the port register, otherwise the buffered pin input state is read. |
| 3<br>PTS | **Port S general purpose input/output data**—Data Register<br>Port S pin 3 is associated with the TXD signal of the SCI1 module.<br>When not used with the alternative function, this pin can be used as general purpose I/O.<br>If the associated data direction bit of this pin is set to 1, a read returns the value of the port register, otherwise the buffered pin input state is read. |
| 2<br>PTS | **Port S general purpose input/output data**—Data Register<br>Port S bits 2 is associated with the RXD signal of the SCI1 module.<br>When not used with the alternative function, this pin can be used as general purpose I/O.<br>If the associated data direction bit of this pin is set to 1, a read returns the value of the port register, otherwise the buffered pin input state is read. |

**Table 2-26. PTS Register Field Descriptions (continued)**

| Field | Description |
|---|---|
| 1<br>PTS | **Port S general purpose input/output data**—Data Register<br>Port S pin 3 is associated with the TXD signal of the SCI0 module.<br>When not used with the alternative function, this pin can be used as general purpose I/O.<br>If the associated data direction bit of this pin is set to 1, a read returns the value of the port register, otherwise the buffered pin input state is read. |
| 0<br>PTS | **Port S general purpose input/output data**—Data Register<br>Port S bits 2 is associated with the RXD signal of the SCI0 module.<br>When not used with the alternative function, this pin can be used as general purpose I/O.<br>If the associated data direction bit of this pin is set to 1, a read returns the value of the port register, otherwise the buffered pin input state is read. |

## 2.3.30 Port S Input Register (PTIS)

Address 0x0249 Access: User read(1)

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | PTIS7 | PTIS6 | PTIS5 | PTIS4 | PTIS3 | PTIS2 | PTIS1 | PTIS0 |
| W | | | | | | | | |
| Reset | u | u | u | u | u | u | u | u |

= Unimplemented or Reserved u = Unaffected by reset

**Figure 2-28. Port S Input Register (PTIS)**

1. Read: Anytime.
   Write:Never, writes to this register have no effect.

**Table 2-27. PTIS Register Field Descriptions**

| Field | Description |
|---|---|
| 7-0<br>PTIS | **Port S input data**—<br>This register always reads back the buffered state of the associated pins. This can also be used to detect overload or short circuit conditions on output pins. |

## 2.3.31 Port S Data Direction Register (DDRS)

Address 0x024A Access: User read/write(1)

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | DDRS7 | DDRS6 | DDRS5 | DDRS4 | DDRS3 | DDRS2 | DDRS1 | DDRS0 |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 2-29. Port S Data Direction Register (DDRS)**

1. Read: Anytime.
   Write: Anytime.

**Table 2-28. DDRS Register Field Descriptions**

| Field | Description |
|---|---|
| 7-0<br>DDRS | **Port S data direction**—<br>This register controls the data direction of pins 7 through 0.This register configures each Port S pin as either input or output.<br>If SPI0 is enabled, the SPI0 determines the pin direction. *Refer to SPI section for details*.<br>If the associated SCI transmit or receive channel is enabled this register has no effect on the pins. The pin is forced to be an output if a SCI transmit channel is enabled, it is forced to be an input if the SCI receive channel is enabled. The data direction bits revert to controlling the I/O direction of a pin when the associated channel is disabled.<br>1 Associated pin is configured as output.<br>0 Associated pin is configured as input. |

**NOTE**

Due to internal synchronization circuits, it can take up to 2 bus clock cycles until the correct value is read on PTS or PTIS registers, when changing the DDRS register.

## 2.3.32 Port S Reduced Drive Register (RDRS)

Address 0x024B — Access: User read/write(1)

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | RDRS7 | RDRS6 | RDRS5 | RDRS4 | RDRS3 | RDRS2 | RDRS1 | RDRS0 |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 2-30. Port S Reduced Drive Register (RDRS)**

1. Read: Anytime.
   Write: Anytime.

**Table 2-29. RDRS Register Field Descriptions**

| Field | Description |
|---|---|
| 7-0<br>RDRS | **Port S reduced drive**—Select reduced drive for outputs<br>This register configures the drive strength of output pins 7 through 0 as either full or reduced independent of the function used on the pins. If a pin is used as input this bit has no effect.<br>1 Reduced drive selected (approx. 1/5 of the full drive strength).<br>0 Full drive strength enabled. |

## 2.3.33 Port S Pull Device Enable Register (PERS)

Address 0x024C Access: User read/write(1)

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | PERS7 | PERS6 | PERS5 | PERS4 | PERS3 | PERS2 | PERS1 | PERS0 |
| Reset | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

**Figure 2-31. Port S Pull Device Enable Register (PERS)**

1. Read: Anytime.
   Write: Anytime.

**Table 2-30. PERS Register Field Descriptions**

| Field | Description |
|---|---|
| 7-0<br>PERS | **Port S pull device enable**—Enable pull devices on input pins<br>These bits configure whether a pull device is activated, if the associated pin is used as an input. This bit has no effect if the pin is used as an output. Out of reset all pull devices are enabled.<br>1 Pull device enabled.<br>0 Pull device disabled. |

## 2.3.34 Port S Polarity Select Register (PPSS)

Address 0x024D Access: User read/write(1)

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | PPSS7 | PPSS6 | PPSS5 | PPSS4 | PPSS3 | PPSS2 | PPSS1 | PPSS0 |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 2-32. Port S Polarity Select Register (PPSS)**

1. Read: Anytime.
   Write: Anytime.

**Table 2-31. PPSS Register Field Descriptions**

| Field | Description |
|---|---|
| 7-0<br>PPSS | **Port S pull device select**—Determine pull device polarity on input pins<br>This register selects whether a pull-down or a pull-up device is connected to the pin.<br>1 A pull-down device is connected to the associated pin, if enabled and if the pin is used as input.<br>0 A pull-up device is connected to the associated pin, if enabled and if the pin is used as input. |

## 2.3.35 Port S Wired-Or Mode Register (WOMS)

Address 0x024E Access: User read/write(1)

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | WOMS7 | WOMS6 | WOMS5 | WOMS4 | WOMS3 | WOMS2 | WOMS1 | WOMS0 |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 2-33. Port S Wired-Or Mode Register (WOMS)**

1. Read: Anytime.
   Write: Anytime.

**Table 2-32. WOMS Register Field Descriptions**

| Field | Description |
|---|---|
| 7-0<br>WOMS | **Port S wired-or mode**—Enable wired-or functionality<br>This register configures the output pins as wired-or independent of the function used on the pins. If enabled the output is driven active low only (open-drain). A logic level of “1” is not driven.This allows a multipoint connection of several serial modules. These bits have no influence on pins used as inputs.<br>1 Output buffers operate as open-drain outputs.<br>0 Output buffers operate as push-pull outputs. |

## 2.3.36 PIM Reserved Register

Address 0x024F Access: User read(1)

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| W | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented or Reserved u = Unaffected by reset

**Figure 2-34. PIM Reserved Register**

1. Read: Always reads 0x00
   Write: Unimplemented

## 2.3.37 Port M Data Register (PTM)

Address 0x0250 Access: User read/write(1)

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | PTM7 | PTM6 | PTM5 | PTM4 | PTM3 | PTM2 | PTM1 | PTM0 |
| Altern. Function | TXCAN3 | RXCAN3 | TXCAN2 | RXCAN2 | TXCAN1 | RXCAN1 | TXCAN0 | RXCAN0 |
| | — | — | (TXCAN0) | (RXCAN0) | (TXCAN0) | (RXCAN0) | — | — |
| | (TXCAN4) | (RXCAN4) | (TXCAN4) | (RXCAN4) | — | — | — | — |
| | — | — | (SCK0) | (MOSI0) | ($\overline{SS0}$) | (MISO0) | — | — |
| | TXD3 | RXD3 | — | — | — | — | — | — |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 2-35. Port M Data Register (PTM)**

1. Read: Anytime.
   Write: Anytime.

**Table 2-33. PTM Register Field Descriptions**

| Field | Description |
|---|---|
| 7-6<br>PTM | **Port M general purpose input/output data**—Data Register<br>Port M pins 7 and 6 are associated with TXCAN and RXCAN signals of CAN3 and the routed CAN4, as well as with TXD and RXD signals of SCI3, respectively.<br>The CAN3 function takes precedence over the CAN4, SCI3 and the general purpose I/O function if the CAN3 module is enabled. The CAN4 function takes precedence over the SCI3 and the general purpose I/O function if the CAN4 module is enabled. The SCI3 function takes precedence over the general purpose I/O function if the SCI3 module is enabled.<br>When not used with the alternative function, this pin can be used as general purpose I/O.<br>If the associated data direction bit of this pin is set to 1, a read returns the value of the port register, otherwise the buffered pin input state is read. |
| 5<br>PTM | **Port M general purpose input/output data**—Data Register<br>Port M pin 5 is associated with the TXCAN signal of CAN2 and the routed CAN4 and CAN0, as well as with SCK signals of SPI0.<br>The CAN2 function takes precedence over the routed CAN0, routed CAN4, the routed SPI0 and the general purpose I/O function if the CAN2 module is enabled. The routed CAN0 function takes precedence over the routed CAN4, the routed SPI0 and the general purpose I/O function if the routed CAN0 module is enabled. The routed CAN4 function takes precedence over the routed SPI0 and general purpose I/O function if the routed CAN4 module is enabled. The routed SPI0 function takes precedence of the general purpose I/O function if the routed SPI0 is enabled.<br>When not used with the alternative function, this pin can be used as general purpose I/O.<br>If the associated data direction bit of this pin is set to 1, a read returns the value of the port register, otherwise the buffered pin input state is read. |

**Table 2-33. PTM Register Field Descriptions (continued)**

| Field | Description |
|---|---|
| 4<br>PTM | **Port M general purpose input/output data**—Data Register<br>Port M pin 4 is associated with the RXCAN signal of CAN2 and the routed CAN4 and CAN0, as well as with MOSI signals of SPI0.<br>The CAN2 function takes precedence over the routed CAN0, routed CAN4, the routed SPI0 and the general purpose I/O function if the CAN2 module is enabled. The routed CAN0 function takes precedence over the routed CAN4, the routed SPI0 and the general purpose I/O function if the routed CAN0 module is enabled. The routed CAN4 function takes precedence over the routed SPI0 and general purpose I/O function if the routed CAN4 module is enabled. The routed SPI0 function takes precedence of the general purpose I/O function if the routed SPI0 is enabled.<br>When not used with the alternative function, this pin can be used as general purpose I/O.<br>If the associated data direction bit of this pin is set to 1, a read returns the value of the port register, otherwise the buffered pin input state is read. |
| 3<br>PTM | **Port M general purpose input/output data**—Data Register<br>Port M pin 5 is associated with the TXCAN signal of CAN1 and the routed CAN0, as well as with $\overline{SS0}$ signals of SPI0.<br>The CAN1 function takes precedence over the routed CAN0, the routed SPI0 and the general purpose I/O function if the CAN1 module is enabled. The routed CAN0 function takes precedence over the routed SPI0 and the general purpose I/O function if the routed CAN0 module is enabled. The routed SPI0 function takes precedence of the general purpose I/O function if the routed SPI0 is enabled.<br>When not used with the alternative function, this pin can be used as general purpose I/O.<br>If the associated data direction bit of this pin is set to 1, a read returns the value of the port register, otherwise the buffered pin input state is read. |
| 2<br>PTM | **Port M general purpose input/output data**—Data Register<br>Port M pin 4 is associated with the RXCAN signal of CAN1 and the routed CAN0, as well as with MISO signals of SPI0.<br>The CAN1 function takes precedence over the routed CAN0, the routed SPI0 and the general purpose I/O function if the CAN1 module is enabled. The routed CAN0 function takes precedence over the routed SPI0 and the general purpose I/O function if the routed CAN0 module is enabled. The routed SPI0 function takes precedence of the general purpose I/O function if the routed SPI0 is enabled.<br>When not used with the alternative function, this pin can be used as general purpose I/O.<br>If the associated data direction bit of this pin is set to 1, a read returns the value of the port register, otherwise the buffered pin input state is read. |
| 1-0<br>PTM | **Port M general purpose input/output data**—Data Register<br>Port M pins 1 and 0 are associated with TXCAN and RXCAN signals of CAN0, respectively.<br>When not used with the alternative function, this pin can be used as general purpose I/O.<br>If the associated data direction bit of this pin is set to 1, a read returns the value of the port register, otherwise the buffered pin input state is read. |

## 2.3.38 Port M Input Register (PTIM)

Address 0x0251 Access: User read(1)

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | PTIM7 | PTIM6 | PTIM5 | PTIM4 | PTIM3 | PTIM2 | PTIM1 | PTIM0 |
| W | | | | | | | | |
| Reset | u | u | u | u | u | u | u | u |

= Unimplemented or Reserved u = Unaffected by reset

**Figure 2-36. Port M Input Register (PTIM)**

1. Read: Anytime.
   Write:Never, writes to this register have no effect.

**Table 2-34. PTIM Register Field Descriptions**

| Field | Description |
|---|---|
| 7-0<br>PTIM | **Port M input data**—<br>This register always reads back the buffered state of the associated pins. This can also be used to detect overload or short circuit conditions on output pins. |

## 2.3.39 Port M Data Direction Register (DDRM)

Address 0x0252 Access: User read/write(1)

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | DDRM7 | DDRM6 | DDRM5 | DDRM4 | DDRM3 | DDRM2 | DDRM1 | DDRM0 |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 2-37. Port M Data Direction Register (DDRM)**

1. Read: Anytime.
   Write: Anytime.

**Table 2-35. DDRM Register Field Descriptions**

| Field | Description |
|---|---|
| 7<br>DDRM | **Port M data direction**—<br>This register controls the data direction of pin 7.<br>The enabled CAN3, routed CAN4, or routed SCI3 forces the I/O state to be an output. In those cases the data direction bits will not change. The DDRM bits revert to controlling the I/O direction of a pin when the associated peripheral module is disabled.<br>1 Associated pin is configured as output.<br>0 Associated pin is configured as input. |
| 6<br>DDRM | **Port M data direction**—<br>This register controls the data direction of pin 6.<br>The enabled CAN3, routed CAN4, or routed SCI3 forces the I/O state to be an input. In those cases the data direction bits will not change. The DDRM bits revert to controlling the I/O direction of a pin when the associated peripheral module is disabled.<br>1 Associated pin is configured as output.<br>0 Associated pin is configured as input. |
| 5<br>DDRM | **Port M data direction**—<br>This register controls the data direction of pin 5.<br>The enabled CAN2, routed CAN0, or routed CAN4 forces the I/O state to be an output. Depending on the configuration of the enabled routed SPI0 this pin will be forced to be input or output. In those cases the data direction bits will not change. The DDRM bits revert to controlling the I/O direction of a pin when the associated peripheral module is disabled.<br>1 Associated pin is configured as output.<br>0 Associated pin is configured as input. |

**Table 2-35. DDRM Register Field Descriptions (continued)**

| Field | Description |
|---|---|
| 4<br>DDRM | **Port M data direction**—<br>This register controls the data direction of pin 4.<br>The enabled CAN2, routed CAN0, or routed CAN4 forces the I/O state to be an input. Depending on the configuration of the enabled routed SPI0 this pin will be forced to be input or output.In those cases the data direction bits will not change. The DDRM bits revert to controlling the I/O direction of a pin when the associated peripheral module is disabled.<br>1 Associated pin is configured as output.<br>0 Associated pin is configured as input. |
| 3<br>DDRM | **Port M data direction**—<br>This register controls the data direction of pin 3.<br>The enabled CAN1 or routed CAN0 forces the I/O state to be an output. Depending on the configuration of the enabled routed SPI0 this pin will be forced to be input or output. In those cases the data direction bits will not change. The DDRM bits revert to controlling the I/O direction of a pin when the associated peripheral module is disabled.<br>1 Associated pin is configured as output.<br>0 Associated pin is configured as input. |
| 2<br>DDRM | **Port M data direction**—<br>This register controls the data direction of pin 2.<br>The enabled CAN1 or routed CAN0 forces the I/O state to be an input. Depending on the configuration of the enabled routed SPI0 this pin will be forced to be input or output.In those cases the data direction bits will not change. The DDRM bits revert to controlling the I/O direction of a pin when the associated peripheral module is disabled.<br>1 Associated pin is configured as output.<br>0 Associated pin is configured as input. |
| 1<br>DDRM | **Port M data direction**—<br>This register controls the data direction of pin 1.<br>The enabled CAN0 forces the I/O state to be an output. In those cases the data direction bits will not change. The DDRM bits revert to controlling the I/O direction of a pin when the associated peripheral module is disabled.<br>1 Associated pin is configured as output.<br>0 Associated pin is configured as input. |
| 0<br>DDRM | **Port M data direction**—<br>This register controls the data direction of pin 0.<br>The enabled CAN0 forces the I/O state to be an input. In those cases the data direction bits will not change. The DDRM bits revert to controlling the I/O direction of a pin when the associated peripheral module is disabled.<br>1 Associated pin is configured as output.<br>0 Associated pin is configured as input. |

**NOTE**

Due to internal synchronization circuits, it can take up to 2 bus clock cycles until the correct value is read on PTM or PTIM registers, when changing the DDRM register.

## 2.3.40 Port M Reduced Drive Register (RDRM)

Address 0x0253 Access: User read/write(1)

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | RDRM7 | RDRM6 | RDRM5 | RDRM4 | RDRM3 | RDRM2 | RDRM1 | RDRM0 |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 2-38. Port M Reduced Drive Register (RDRM)**

1. Read: Anytime.
   Write: Anytime.

**Table 2-36. RDRM Register Field Descriptions**

| Field | Description |
|---|---|
| 7-0<br>RDRM | **Port M reduced drive**—Select reduced drive for outputs<br>This register configures the drive strength of Port M output pins 7 through 0 as either full or reduced independent of the function used on the pins. If a pin is used as input this bit has no effect.<br>1 Reduced drive selected (approx. 1/5 of the full drive strength).<br>0 Full drive strength enabled. |

## 2.3.41 Port M Pull Device Enable Register (PERM)

Address 0x0254 Access: User read/write(1)

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | PERM7 | PERM6 | PERM5 | PERM4 | PERM3 | PERM2 | PERM1 | PERM0 |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 2-39. Port M Pull Device Enable Register (PERM)**

1. Read: Anytime.
   Write: Anytime.

**Table 2-37. PERM Register Field Descriptions**

| Field | Description |
|---|---|
| 7-0<br>PERM | **Port M pull device enable**—Enable pull-up devices on input pins<br>These bits configure whether a pull device is activated, if the associated pin is used as an input or wired-or output. This bit has no effect if the pin is used as push-pull output. Out of reset no pull device is enabled.<br>1 Pull device enabled.<br>0 Pull device disabled. |

## 2.3.42 Port M Polarity Select Register (PPSM)

Address 0x0255 Access: User read/write(1)

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | PPSM7 | PPSM6 | PPSM5 | PPSM4 | PPSM3 | PPSM2 | PPSM1 | PPSM0 |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 2-40. Port M Polarity Select Register (PPSM)**

1. Read: Anytime.
   Write: Anytime.

**Table 2-38. PPSM Register Field Descriptions**

| Field | Description |
|---|---|
| 7-0<br>PPSM | **Port M pull device select**—Determine pull device polarity on input pins<br>This register selects whether a pull-down or a pull-up device is connected to the pin. If CAN is active a pull-up device can be activated on the RXCAN[3:0] inputs, but not a pull-down.<br>1 A pull-down device is connected to the associated Port M pin, if enabled by the associated bit in register PERM and if the port is used as a general purpose but not as RXCAN.<br>0 A pull-up device is connected to the associated Port M pin, if enabled by the associated bit in register PERM and if the port is used as general purpose or RXCAN input. |

## 2.3.43 Port M Wired-Or Mode Register (WOMM)

Address 0x0256 Access: User read/write(1)

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | WOMM7 | WOMM6 | WOMM5 | WOMM4 | WOMM3 | WOMM2 | WOMM1 | WOMM0 |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 2-41. Port M Wired-Or Mode Register (WOMM)**

1. Read: Anytime.
   Write: Anytime.

**Table 2-39. WOMM Register Field Descriptions**

| Field | Description |
|---|---|
| 7-0<br>WOMM | **Port M wired-or mode**—Enable wired-or functionality<br>This register configures the output pins as wired-or independent of the function used on the pins. If enabled the output is driven active low only (open-drain). A logic level of “1” is not driven.This allows a multipoint connection of several serial modules. These bits have no influence on pins used as inputs.<br>1 Output buffers operate as open-drain outputs.<br>0 Output buffers operate as push-pull outputs. |

## 2.3.44 Module Routing Register (MODRR)

Address 0x0257 Access: User read/write(1)

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | 0 | MODRR6 | MODRR5 | MODRR4 | MODRR3 | MODRR2 | MODRR1 | MODRR0 |
| W | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented or Reserved

**Figure 2-42. Module Routing Register (MODRR)**

1. Read: Anytime.
   Write: Anytime.

This register configures the re-routing of CAN0, CAN4, SPI0, SPI1, and SPI2 on alternative ports.

**Table 2-40. Module Routing Summary**

| Module | MODRR | | | | | | | Related Pins | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | 6 | 5 | 4 | 3 | 2 | 1 | 0 | | | | |
| | | | | | | | | RXCAN | | TXCAN | |
| CAN0 | x | x | x | x | x | 0 | 0 | PM0 | | PM1 | |
| | x | x | x | x | x | 0 | 1 | PM2 | | PM3 | |
| | x | x | x | x | x | 1 | 0 | PM4 | | PM5 | |
| | x | x | x | x | x | 1 | 1 | PJ6 | | PJ7 | |
| CAN4 | x | x | x | 0 | 0 | x | x | PJ6 | | PJ7 | |
| | x | x | x | 0 | 1 | x | x | PM4 | | PM5 | |
| | x | x | x | 1 | 0 | x | x | PM6 | | PM7 | |
| | x | x | x | 1 | 1 | x | x | Reserved | | | |
| | | | | | | | | MISO | MOSI | SCK | $\overline{SS}$ |
| SPI0 | x | x | 0 | x | x | x | x | PS4 | PS5 | PS6 | PS7 |
| | x | x | 1 | x | x | x | x | PM2 | PM4 | PM5 | PM3 |
| SPI1 | x | 0 | x | x | x | x | x | PP0 | PP1 | PP2 | PP3 |
| | x | 1 | x | x | x | x | x | PH0 | PH1 | PH2 | PH3 |
| SPI2 | 0 | x | x | x | x | x | x | PP4 | PP5 | PP7 | PP6 |
| | 1 | x | x | x | x | x | x | PH4 | PH5 | PH6 | PH7 |

## 2.3.45 Port P Data Register (PTP)

Address 0x0258 Access: User read/write(1)

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | PTP7 | PTP6 | PTP5 | PTP4 | PTP3 | PTP2 | PTP1 | PTP0 |
| Altern. Function | PWM7 | PWM6 | PWM5 | PWM4 | PWM3 | PWM2 | PWM1 | PWM0 |
| | SCK2 | $\overline{SS2}$ | MOSI2 | MISO2 | $\overline{SS1}$ | SCK1 | MOSI1 | MISO1 |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 2-43. Port P Data Register (PTP)**

1. Read: Anytime.
   Write: Anytime.

**Table 2-41. PTP Register Field Descriptions**

| Field | Description |
|---|---|
| 7<br>PTP | **Port P general purpose input/output data**—Data Register<br>Port P pin 6 is associated with the PWM output channel 7 and the SCK signal of SPI2.<br>The PWM function takes precedence over the SPI2 and the general purpose I/O function if the PWM channel 7 is enabled. The SPI2 function takes precedence of the general purpose I/O function if the routed SPI2 is enabled. When not used with the alternative functions, these pins can be used as general purpose I/O.<br>If the associated data direction bits of these pins are set to 1, a read returns the value of the port register, otherwise the buffered pin input state is read. |
| 6<br>PTP | **Port P general purpose input/output data**—Data Register<br>Port P pin 6 is associated with the PWM output channel 6 and the $\overline{SS}$ signal of SPI2.<br>The PWM function takes precedence over the SPI2 and the general purpose I/O function if the PWM channel 6 is enabled. The SPI2 function takes precedence of the general purpose I/O function if the routed SPI2 is enabled. When not used with the alternative functions, these pins can be used as general purpose I/O.<br>If the associated data direction bits of these pins are set to 1, a read returns the value of the port register, otherwise the buffered pin input state is read. |
| 5<br>PTP | **Port P general purpose input/output data**—Data Register<br>Port P pin 5 is associated with the PWM output channel 5 and the MOSI signal of SPI2.<br>The PWM function takes precedence over the SPI2 and the general purpose I/O function if the PWM channel 5 is enabled. The SPI2 function takes precedence of the general purpose I/O function if the routed SPI2 is enabled. When not used with the alternative functions, these pins can be used as general purpose I/O.<br>If the associated data direction bits of these pins are set to 1, a read returns the value of the port register, otherwise the buffered pin input state is read. |
| 4<br>PTP | **Port P general purpose input/output data**—Data Register<br>Port P pin 4 is associated with the PWM output channel 4 and the MISO signal of SPI2.<br>The PWM function takes precedence over the SPI2 and the general purpose I/O function if the PWM channel 4 is enabled. The SPI2 function takes precedence of the general purpose I/O function if the routed SPI2 is enabled. When not used with the alternative functions, these pins can be used as general purpose I/O.<br>If the associated data direction bits of these pins are set to 1, a read returns the value of the port register, otherwise the buffered pin input state is read. |

**Table 2-41. PTP Register Field Descriptions (continued)**

| Field | Description |
|---|---|
| 3<br>PTP | **Port P general purpose input/output data**—Data Register<br>Port P pin 3 is associated with the PWM output channel 3 and the $\overline{SS}$ signal of SPI1.<br>The PWM function takes precedence over the SPI1 and the general purpose I/O function if the PWM channel 3 is enabled. The SPI1 function takes precedence of the general purpose I/O function if the routed SPI1 is enabled.<br>When not used with the alternative functions, these pins can be used as general purpose I/O.<br>If the associated data direction bits of these pins are set to 1, a read returns the value of the port register, otherwise the buffered pin input state is read. |
| 2<br>PTP | **Port P general purpose input/output data**—Data Register<br>Port P pin 2 is associated with the PWM output channel 2 and the SCK signal of SPI1.<br>The PWM function takes precedence over the SPI1 and the general purpose I/O function if the PWM channel 2 is enabled. The SPI1 function takes precedence of the general purpose I/O function if the routed SPI1 is enabled.<br>When not used with the alternative functions, these pins can be used as general purpose I/O.<br>If the associated data direction bits of these pins are set to 1, a read returns the value of the port register, otherwise the buffered pin input state is read. |
| 1<br>PTP | **Port P general purpose input/output data**—Data Register<br>Port P pin 1 is associated with the PWM output channel 1 and the MOSI signal of SPI1.<br>The PWM function takes precedence over the SPI1 and the general purpose I/O function if the PWM channel 1 is enabled. The SPI1 function takes precedence of the general purpose I/O function if the routed SPI1 is enabled.<br>When not used with the alternative functions, these pins can be used as general purpose I/O.<br>If the associated data direction bits of these pins are set to 1, a read returns the value of the port register, otherwise the buffered pin input state is read. |
| 0<br>PTP | **Port P general purpose input/output data**—Data Register<br>Port P pin 0 is associated with the PWM output channel 0 and the MISO signal of SPI1.<br>The PWM function takes precedence over the SPI1 and the general purpose I/O function if the PWM channel 0 is enabled. The SPI1 function takes precedence of the general purpose I/O function if the routed SPI1 is enabled.<br>When not used with the alternative functions, these pins can be used as general purpose I/O.<br>If the associated data direction bits of these pins are set to 1, a read returns the value of the port register, otherwise the buffered pin input state is read. |

## 2.3.46 Port P Input Register (PTIP)

Address 0x0259 Access: User read(1)

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | PTIP7 | PTIP6 | PTIP5 | PTIP4 | PTIP3 | PTIP2 | PTIP1 | PTIP0 |
| W | | | | | | | | |
| Reset | u | u | u | u | u | u | u | u |

= Unimplemented or Reserved u = Unaffected by reset

**Figure 2-44. Port P Input Register (PTIP)**

1. Read: Anytime.
   Write:Never, writes to this register have no effect.

**Table 2-42. PTIP Register Field Descriptions**

| Field | Description |
|---|---|
| 7-0<br>PTIP | **Port P input data**—<br>This register always reads back the buffered state of the associated pins. This can also be used to detect overload or short circuit conditions on output pins. |

## 2.3.47 Port P Data Direction Register (DDRP)

Address 0x025A — Access: User read/write(1)

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | DDRP7 | DDRP6 | DDRP5 | DDRP4 | DDRP3 | DDRP2 | DDRP1 | DDRP0 |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 2-45. Port P Data Direction Register (DDRP)**

1. Read: Anytime.
   Write: Anytime.

**Table 2-43. DDRP Register Field Descriptions**

| Field | Description |
|---|---|
| 7<br>DDRP | **Port P data direction**—<br>This register controls the data direction of pin 7.<br>The enabled PWM channel 7 forces the I/O state to be an output. If the PWM shutdown feature is enabled this pin is forced to be an input. In these cases the data direction bit will not change.<br>1 Associated pin is configured as output.<br>0 Associated pin is configured as input. |
| 6-0<br>DDRP | **Port P data direction**—<br>The PWM forces the I/O state to be an output for each port line associated with an enabled PWM6-0 channel. In this case the data direction bit will not change.<br>1 Associated pin is configured as output.<br>0 Associated pin is configured as input. |

**NOTE**

Due to internal synchronization circuits, it can take up to 2 bus clock cycles until the correct value is read on PTP or PTIP registers, when changing the DDRP register.

## 2.3.48 Port P Reduced Drive Register (RDRP)

Address 0x025B Access: User read/write(1)

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | RDRP7 | RDRP6 | RDRP5 | RDRP4 | RDRP3 | RDRP2 | RDRP1 | RDRP0 |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 2-46. Port P Reduced Drive Register (RDRP)**

1. Read: Anytime.
   Write: Anytime.

**Table 2-44. RDRP Register Field Descriptions**

| Field | Description |
|---|---|
| 7-0<br>RDRP | **Port P reduced drive**—Select reduced drive for outputs<br>This register configures the drive strength of output pins 7 through 0 as either full or reduced independent of the function used on the pins. If a pin is used as input this bit has no effect.<br>1 Reduced drive selected (approx. 1/5 of the full drive strength).<br>0 Full drive strength enabled. |

## 2.3.49 Port P Pull Device Enable Register (PERP)

Address 0x025C Access: User read/write(1)

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | PPSP7 | PPSP6 | PPSP5 | PPSP4 | PPSP3 | PPSP2 | PPSP1 | PPSP0 |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 2-47. Port P Pull Device Enable Register (PERP)**

1. Read: Anytime.
   Write: Anytime.

**Table 2-45. PERP Register Field Descriptions**

| Field | Description |
|---|---|
| 7-0<br>PERP | **Port P pull device enable**—Enable pull devices on input pins<br>These bits configure whether a pull device is activated, if the associated pin is used as an input. This bit has no effect if the pin is used as an output. Out of reset no pull device is enabled.<br>1 Pull device enabled.<br>0 Pull device disabled. |

## 2.3.50 Port P Polarity Select Register (PPSP)

Address 0x025D Access: User read/write(1)

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | PPSP7 | PPSP6 | PPSP5 | PPSP4 | PPSP3 | PPSP2 | PPSP1 | PPSP0 |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 2-48. Port P Polarity Select Register (PPSP)**

1. Read: Anytime.
   Write: Anytime.

**Table 2-46. PPSP Register Field Descriptions**

| Field | Description |
|---|---|
| 7-0<br>PPSP | **Port P pull device select**—Determine pull device polarity on input pins<br>This register serves a dual purpose by selecting the polarity of the active interrupt edge as well as selecting a pull-up or pull-down device if enabled.<br>1 A rising edge on the associated Port P pin sets the associated flag bit in the PIFP register. A pull-down device is connected to the associated Port P pin, if enabled by the associated bit in register PERP and if the port is used as input.<br>0 A falling edge on the associated Port P pin sets the associated flag bit in the PIFP register.A pull-up device is connected to the associated Port P pin, if enabled by the associated bit in register PERP and if the port is used as input. |

## 2.3.51 Port P Interrupt Enable Register (PIEP)

Read: Anytime.

Address 0x025E Access: User read/write(1)

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | PIEP7 | PIEP6 | PIEP5 | PIEP4 | PIEP3 | PIEP2 | PIEP1 | PIEP0 |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 2-49. Port P Interrupt Enable Register (PIEP)**

1. Read: Anytime.
   Write: Anytime.

**Table 2-47. PPSP Register Field Descriptions**

| Field | Description |
|---|---|
| 7-0<br>PIEP | **Port P interrupt enable**—<br>This register disables or enables on a per-pin basis the edge sensitive external interrupt associated with Port P.<br>1 Interrupt is enabled.<br>0 Interrupt is disabled (interrupt flag masked). |

## 2.3.52 Port P Interrupt Flag Register (PIFP)

Address 0x025F Access: User read/write(1)

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | PIFP7 | PIFP6 | PIFP5 | PIFP4 | PIFP3 | PIFP2 | PIFP1 | PIFP0 |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 2-50. Port P Interrupt Flag Register (PIFP)**

1. Read: Anytime.
   Write: Anytime.

**Table 2-48. PPSP Register Field Descriptions**

| Field | Description |
|---|---|
| 7-0<br>PIFP | **Port P interrupt flag—**<br>Each flag is set by an active edge on the associated input pin. This could be a rising or a falling edge based on the state of the PPSP register. To clear this flag, write logic level 1 to the corresponding bit in the PIFP register. Writing a 0 has no effect.<br>1 Active edge on the associated bit has occurred (an interrupt will occur if the associated enable bit is set).<br>0 No active edge pending. |

## 2.3.53 Port H Data Register (PTH)

Address 0x0260 Access: User read/write(1)

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | PTH7 | PTH6 | PTH5 | PTH4 | PTH3 | PTH2 | PTH1 | PTH0 |
| Altern. Function | $\overline{SS2}$ | SCK2 | MOSI2 | MISO2 | $\overline{SS1}$ | SCK1 | MOSI1 | MISO1 |
| | TXD5 | RXD5 | TXD4 | RXD4 | TXD7 | RXD7 | TXD6 | RXD6 |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 2-51. Port H Data Register (PTH)**

1. Read: Anytime.
   Write: Anytime.

**Table 2-49. PTH Register Field Descriptions**

| Field | Description |
|---|---|
| 7<br>PTH | **Port H general purpose input/output data**—Data Register<br>Port H pin 7 is associated with the TXD signal of the SCI5 module and the $\overline{SS}$ signal of the routed SPI2.<br>The routed SPI2 function takes precedence over the SCI5 and the general purpose I/O function if the routed SPI2 module is enabled. The SCI5 function takes precedence over the general purpose I/O function if the SCI5 is enabled.<br>When not used with the alternative function, this pin can be used as general purpose I/O.<br>If the associated data direction bit of this pin is set to 1, a read returns the value of the port register, otherwise the buffered pin input state is read. |
| 6<br>PTH | **Port H general purpose input/output data**—Data Register<br>Port H pin 6 is associated with the RXD signal of the SCI5 module and the SCK signal of the routed SPI2.<br>The routed SPI2 function takes precedence over the SCI5 and the general purpose I/O function if the routed SPI2 module is enabled. The SCI5 function takes precedence over the general purpose I/O function if the SCI5 is enabled.<br>When not used with the alternative function, this pin can be used as general purpose I/O.<br>If the associated data direction bit of this pin is set to 1, a read returns the value of the port register, otherwise the buffered pin input state is read. |
| 5<br>PTH | **Port H general purpose input/output data**—Data Register<br>Port H pin 5 is associated with the TXD signal of the SCI4 module and the MOSI signal of the routed SPI2.<br>The routed SPI2 function takes precedence over the SCI4 and the general purpose I/O function if the routed SPI2 module is enabled. The SCI4 function takes precedence over the general purpose I/O function if the SCI4 is enabled.<br>When not used with the alternative function, this pin can be used as general purpose I/O.<br>If the associated data direction bit of this pin is set to 1, a read returns the value of the port register, otherwise the buffered pin input state is read. |
| 4<br>PTH | **Port H general purpose input/output data**—Data Register<br>Port H pin 4 is associated with the RXD signal of the SCI4 module and the MISO signal of the routed SPI2.<br>The routed SPI2 function takes precedence over the SCI4 and the general purpose I/O function if the routed SPI2 module is enabled. The SCI4 function takes precedence over the general purpose I/O function if the SCI4 is enabled.<br>When not used with the alternative function, this pin can be used as general purpose I/O.<br>If the associated data direction bit of this pin is set to 1, a read returns the value of the port register, otherwise the buffered pin input state is read. |
| 3<br>PTH | **Port H general purpose input/output data**—Data Register<br>Port H pin 3 is associated with the TXD signal of the SCI7 module and the $\overline{SS}$ signal of the routed SPI1.<br>The routed SPI1 function takes precedence over the SCI7 and the general purpose I/O function if the routed SPI1 module is enabled. The SCI7 function takes precedence over the general purpose I/O function if the SCI7 is enabled.<br>When not used with the alternative function, this pin can be used as general purpose I/O.<br>If the associated data direction bit of this pin is set to 1, a read returns the value of the port register, otherwise the buffered pin input state is read. |
| 2<br>PTH | **Port H general purpose input/output data**—Data Register<br>Port H pin 2 is associated with the RXD signal of the SCI7 module and the SCK signal of the routed SPI1.<br>The routed SPI1 function takes precedence over the SCI7 and the general purpose I/O function if the routed SPI1 module is enabled. The SCI7 function takes precedence over the general purpose I/O function if the SCI7 is enabled.<br>When not used with the alternative function, this pin can be used as general purpose I/O.<br>If the associated data direction bit of this pin is set to 1, a read returns the value of the port register, otherwise the buffered pin input state is read. |

**Table 2-49. PTH Register Field Descriptions (continued)**

| Field | Description |
|---|---|
| 1<br>PTH | **Port H general purpose input/output data**—Data Register<br>Port H pin 1 is associated with the TXD signal of the SCI6 module and the MOSI signal of the routed SPI1.<br>The routed SPI1 function takes precedence over the SCI6 and the general purpose I/O function if the routed SPI1 module is enabled. The SCI6 function takes precedence over the general purpose I/O function if the SCI6 is enabled.<br>When not used with the alternative function, this pin can be used as general purpose I/O.<br>If the associated data direction bit of this pin is set to 1, a read returns the value of the port register, otherwise the buffered pin input state is read. |
| 0<br>PTH | **Port H general purpose input/output data**—Data Register<br>Port H pin 0 is associated with the RXD signal of the SCI6 module and the MISO signal of the routed SPI1.<br>The routed SPI1 function takes precedence over the SCI6 and the general purpose I/O function if the routed SPI1 module is enabled. The SCI6 function takes precedence over the general purpose I/O function if the SCI6 is enabled.<br>When not used with the alternative function, this pin can be used as general purpose I/O.<br>If the associated data direction bit of this pin is set to 1, a read returns the value of the port register, otherwise the buffered pin input state is read. |

## 2.3.54 Port H Input Register (PTIH)

Address 0x0261 Access: User read(1)

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | PTIH7 | PTIH6 | PTIH5 | PTIH4 | PTIH3 | PTIH2 | PTIH1 | PTIH0 |
| W | | | | | | | | |
| Reset | u | u | u | u | u | u | u | u |

= Unimplemented or Reserved u = Unaffected by reset

**Figure 2-52. Port H Input Register (PTIH)**

1. Read: Anytime.
   Write:Never, writes to this register have no effect.

**Table 2-50. PTIH Register Field Descriptions**

| Field | Description |
|---|---|
| 7-0<br>PTIH | **Port H input data**—<br>This register always reads back the buffered state of the associated pins. This can also be used to detect overload or short circuit conditions on output pins. |

## 2.3.55 Port H Data Direction Register (DDRH)

Address 0x0262 Access: User read/write(1)

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | DDRH7 | DDRH6 | DDRH5 | DDRH4 | DDRH3 | DDRH2 | DDRH1 | DDRH0 |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 2-53. Port H Data Direction Register (DDRH)**

1. Read: Anytime.
   Write: Anytime.

**Table 2-51. DDRH Register Field Descriptions**

| Field | Description |
|---|---|
| 7<br>DDRH | **Port H data direction**—<br>This register controls the data direction of pin 7.<br>The enabled SCI5 forces the I/O state to be an output. Depending on the configuration of the enabled routed SPI2 this pin will be forced to be input or output. In those cases the data direction bits will not change. The DDRM bits revert to controlling the I/O direction of a pin when the associated peripheral module is disabled.<br>1 Associated pin is configured as output.<br>0 Associated pin is configured as input. |
| 6<br>DDRH | **Port H data direction**—<br>This register controls the data direction of pin 6.<br>The enabled SCI5 forces the I/O state to be an input. Depending on the configuration of the enabled routed SPI2 this pin will be forced to be input or output. In those cases the data direction bits will not change. The DDRM bits revert to controlling the I/O direction of a pin when the associated peripheral module is disabled.<br>1 Associated pin is configured as output.<br>0 Associated pin is configured as input. |
| 5<br>DDRH | **Port H data direction**—<br>This register controls the data direction of pin 5.<br>The enabled SCI4 forces the I/O state to be an output. Depending on the configuration of the enabled routed SPI2 this pin will be forced to be input or output. In those cases the data direction bits will not change. The DDRM bits revert to controlling the I/O direction of a pin when the associated peripheral module is disabled.<br>1 Associated pin is configured as output.<br>0 Associated pin is configured as input. |
| 4<br>DDRH | **Port H data direction**—<br>This register controls the data direction of pin 4.<br>The enabled SCI4 forces the I/O state to be an input. Depending on the configuration of the enabled routed SPI2 this pin will be forced to be input or output. In those cases the data direction bits will not change. The DDRM bits revert to controlling the I/O direction of a pin when the associated peripheral module is disabled.<br>1 Associated pin is configured as output.<br>0 Associated pin is configured as input. |
| 3<br>DDRH | **Port H data direction**—<br>This register controls the data direction of pin 3.<br>The enabled SCI7 forces the I/O state to be an output. Depending on the configuration of the enabled routed SPI1 this pin will be forced to be input or output. In those cases the data direction bits will not change. The DDRM bits revert to controlling the I/O direction of a pin when the associated peripheral module is disabled.<br>1 Associated pin is configured as output.<br>0 Associated pin is configured as input. |
| 2<br>DDRH | **Port H data direction**—<br>This register controls the data direction of pin 2.<br>The enabled SCI7 forces the I/O state to be an input. Depending on the configuration of the enabled routed SPI1 this pin will be forced to be input or output. In those cases the data direction bits will not change. The DDRM bits revert to controlling the I/O direction of a pin when the associated peripheral module is disabled.<br>1 Associated pin is configured as output.<br>0 Associated pin is configured as input. |

**Table 2-51. DDRH Register Field Descriptions (continued)**

| Field | Description |
|---|---|
| 1<br>DDRH | **Port H data direction**—<br>This register controls the data direction of pin 1.<br>The enabled SCI6 forces the I/O state to be an output. Depending on the configuration of the enabled routed SPI1 this pin will be forced to be input or output. In those cases the data direction bits will not change. The DDRM bits revert to controlling the I/O direction of a pin when the associated peripheral module is disabled.<br>1 Associated pin is configured as output.<br>0 Associated pin is configured as input. |
| 0<br>DDRH | **Port H data direction**—<br>This register controls the data direction of pin 0.<br>The enabled SCI6 forces the I/O state to be an input. Depending on the configuration of the enabled routed SPI1 this pin will be forced to be input or output. In those cases the data direction bits will not change. The DDRM bits revert to controlling the I/O direction of a pin when the associated peripheral module is disabled.<br>1 Associated pin is configured as output.<br>0 Associated pin is configured as input. |

**NOTE**

Due to internal synchronization circuits, it can take up to 2 bus clock cycles until the correct value is read on PTH or PTIH registers, when changing the DDRH register.

## 2.3.56 Port H Reduced Drive Register (RDRH)

Address 0x0263 Access: User read/write(1)

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | RDRH7 | RDRH6 | RDRH5 | RDRH4 | RDRH3 | RDRH2 | RDRH1 | RDRH0 |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 2-54. Port H Reduced Drive Register (RDRH)**

1. Read: Anytime.
   Write: Anytime.

**Table 2-52. RDRH Register Field Descriptions**

| Field | Description |
|---|---|
| 7-0<br>RDRH | **Port H reduced drive**—Select reduced drive for outputs<br>This register configures the drive strength of output pins 7 through 0 as either full or reduced independent of the function used on the pins. If a pin is used as input this bit has no effect.<br>1 Reduced drive selected (approx. 1/5 of the full drive strength).<br>0 Full drive strength enabled. |

## 2.3.57 Port H Pull Device Enable Register (PERH)

Address 0x0264 Access: User read/write(1)

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | PERH7 | PERH6 | PERH5 | PERH4 | PERH3 | PERH2 | PERH1 | PERH0 |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 2-55. Port H Pull Device Enable Register (PERH)**

1. Read: Anytime.
   Write: Anytime.

**Table 2-53. PERH Register Field Descriptions**

| Field | Description |
|---|---|
| 7-0<br>PERH | **Port H pull device enable**—Enable pull devices on input pins<br>These bits configure whether a pull device is activated, if the associated pin is used as an input. This bit has no effect if the pin is used as an output. Out of reset no pull device is enabled.<br>1 Pull device enabled.<br>0 Pull device disabled. |

## 2.3.58 Port H Polarity Select Register (PPSH)

Address 0x0265 Access: User read/write(1)

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | PPSH7 | PPSH6 | PPSH5 | PPSH4 | PPSH3 | PPSH2 | PPSH1 | PPSH0 |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 2-56. Port H Polarity Select Register (PPSH)**

1. Read: Anytime.
   Write: Anytime.

**Table 2-54. PPSH Register Field Descriptions**

| Field | Description |
|---|---|
| 7-0<br>PPSH | **Port H pull device select**—Determine pull device polarity on input pins<br>This register serves a dual purpose by selecting the polarity of the active interrupt edge as well as selecting a pull-up or pull-down device if enabled.<br>1 A rising edge on the associated Port H pin sets the associated flag bit in the PIFH register. A pull-down device is connected to the associated Port H pin, if enabled by the associated bit in register PERH and if the port is used as input.<br>0 A falling edge on the associated Port H pin sets the associated flag bit in the PIFH register.A pull-up device is connected to the associated Port H pin, if enabled by the associated bit in register PERH and if the port is used as input. |

## 2.3.59 Port H Interrupt Enable Register (PIEH)

Read: Anytime.

Address 0x0266 Access: User read/write(1)

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | PIEH7 | PIEH6 | PIEH5 | PIEH4 | PIEH3 | PIEH2 | PIEH1 | PIEH0 |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 2-57. Port H Interrupt Enable Register (PIEH)**

1. Read: Anytime.
   Write: Anytime.

**Table 2-55. PPSP Register Field Descriptions**

| Field | Description |
|---|---|
| 7-0<br>PIEH | **Port H interrupt enable**—<br>This register disables or enables on a per-pin basis the edge sensitive external interrupt associated with Port H.<br>1 Interrupt is enabled.<br>0 Interrupt is disabled (interrupt flag masked). |

## 2.3.60 Port H Interrupt Flag Register (PIFH)

Address 0x0267 Access: User read/write(1)

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | PIFH7 | PIFH6 | PIFH5 | PIFH4 | PIFH3 | PIFH2 | PIFH1 | PIFH0 |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 2-58. Port H Interrupt Flag Register (PIFH)**

1. Read: Anytime.
   Write: Anytime.

**Table 2-56. PPSP Register Field Descriptions**

| Field | Description |
|---|---|
| 7-0<br>PIFH | **Port H interrupt flag**—<br>Each flag is set by an active edge on the associated input pin. This could be a rising or a falling edge based on the state of the PPSH register. To clear this flag, write logic level 1 to the corresponding bit in the PIFH register. Writing a 0 has no effect.<br>1 Active edge on the associated bit has occurred (an interrupt will occur if the associated enable bit is set).<br>0 No active edge pending. |

## 2.3.61 Port J Data Register (PTJ)

Address 0x0268 Access: User read/write(1)

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | PTJ7 | PTJ6 | PTJ5 | PTJ4 | PTJ3 | PTJ2 | PTJ1 | PTJ0 |
| Altern. Function | TXCAN4 | RXCAN4 | — | — | — | — | TXD2 | RXD2 |
| | SCL0 | SDA0 | SCL1 | SDA1 | — | — | — | — |
| | (TXCAN0) | (RXCAN0) | $\overline{CS2}$ | $\overline{CS0}$ | — | $\overline{CS1}$ | — | $\overline{CS3}$ |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 2-59. Port J Data Register (PTJ)**

1. Read: Anytime.
   Write: Anytime.

**Table 2-57. PTJ Register Field Descriptions**

| Field | Description |
|---|---|
| 7-6<br>PTJ | **Port J general purpose input/output data**—Data Register<br>Port J pins 7 and 6 are associated with TXCAN and RXCAN signals of CAN4 and the routed CAN0, as well as with SCL and SDA signals of IIC0, respectively.<br>The CAN4 function takes precedence over the IIC0, the routed CAN0 and the general purpose I/O function if the CAN4 module is enabled. The IIC0 function takes precedence over the routed CAN0 and the general purpose I/O function if the IIC0 is enabled. If the IIC0 module takes precedence the SDA0 and SCL0 outputs are configured as open drain outputs. The routed CAN0 function takes precedence over the general purpose I/O function if the routed CAN0 module is enabled.<br>When not used with the alternative function, this pin can be used as general purpose I/O.<br>If the associated data direction bit of this pin is set to 1, a read returns the value of the port register, otherwise the buffered pin input state is read. |
| 5-4<br>PTJ | **Port J general purpose input/output data**—Data Register<br>This pin is associated with the SCL and SDA signals of IIC1, and with chip select outputs $\overline{CS2}$ and $\overline{CS0}$, respectively. The IIC1 function takes precedence over the chip select and general purpose I/O function if the IIC1 is enabled. The chip selects take precedence over the general purpose I/O. If the IIC1 module takes precedence the SDA1 and SCL1 outputs are configured as open drain outputs. Refer to IIC section for details.<br>When not used with the alternative function, this pin can be used as general purpose I/O.<br>If the associated data direction bit of this pin is set to 1, a read returns the value of the port register, otherwise the buffered pin input state is read. |
| 3<br>PTJ | **Port J general purpose input/output data**—Data Register<br>This pin can be used as general purpose I/O.<br>If the associated data direction bit of this pin is set to 1, a read returns the value of the port register, otherwise the buffered pin input state is read. |
| 2<br>PTJ | **Port J general purpose input/output data**—Data Register<br>This pin is associated with the chip select output signal $\overline{CS2}$.<br>When not used with the alternative function, this pin can be used as general purpose I/O.<br>If the associated data direction bit of this pin is set to 1, a read returns the value of the port register, otherwise the buffered pin input state is read. |

**Table 2-57. PTJ Register Field Descriptions (continued)**

| Field | Description |
|---|---|
| 1<br>PTJ | **Port J general purpose input/output data**—Data Register<br>This pin is associated with the TXD signal of SCI2.<br>When not used with the alternative function, this pin can be used as general purpose I/O.<br>If the associated data direction bit of this pin is set to 1, a read returns the value of the port register, otherwise the buffered pin input state is read. |
| 0<br>PTJ | **Port J general purpose input/output data**—Data Register<br>This pin is associated with the TXD signal of SCI2 and chip select output $\overline{CS3}$. The SCI function takes precedence over the chip select and general purpose I/O function if the SCI2 is enabled. The chip select takes precedence over the general purpose I/O.<br>When not used with the alternative function, this pin can be used as general purpose I/O.<br>If the associated data direction bit of this pin is set to 1, a read returns the value of the port register, otherwise the buffered pin input state is read. |

## 2.3.62 Port J Input Register (PTIJ)

Address 0x0269 Access: User read(1)

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | PTIJ7 | PTIJ6 | PTIJ5 | PTIJ4 | PTIJ3 | PTIJ2 | PTIJ1 | PTIJ0 |
| W | | | | | | | | |
| Reset | u | u | u | u | u | u | u | u |

= Unimplemented or Reserved u = Unaffected by reset

**Figure 2-60. Port J Input Register (PTIJ)**

1. Read: Anytime.
   Write:Never, writes to this register have no effect.

**Table 2-58. PTIJ Register Field Descriptions**

| Field | Description |
|---|---|
| 7-0<br>PTIJ | **Port J input data**—<br>This register always reads back the buffered state of the associated pins. This can also be used to detect overload or short circuit conditions on output pins. |

## 2.3.63 Port J Data Direction Register (DDRJ)

Address 0x026A Access: User read/write(1)

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | DDRJ7 | DDRJ6 | DDRJ5 | DDRJ4 | DDRJ3 | DDRJ2 | DDRJ1 | DDRJ0 |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 2-61. Port J Data Direction Register (DDRJ)**

1. Read: Anytime.
   Write: Anytime.

**Table 2-59. DDRJ Register Field Descriptions**

| Field | Description |
|---|---|
| 7<br>DDRJ | **Port J data direction**—<br>This register controls the data direction of pin 7.<br>The enabled CAN4 or routed CAN0 forces the I/O state to be an output. The enabled IIC0 module forces this pin to be a open drain output. In those cases the data direction bits will not change. The DDRM bits revert to controlling the I/O direction of a pin when the associated peripheral module is disabled.<br>1 Associated pin is configured as output.<br>0 Associated pin is configured as input. |
| 6<br>DDRJ | **Port J data direction**—<br>This register controls the data direction of pin 6.<br>The enabled CAN4 or routed CAN0 forces the I/O state to be an input. The enabled IIC0 module forces this pin to be a open drain output. In those cases the data direction bits will not change. The DDRM bits revert to controlling the I/O direction of a pin when the associated peripheral module is disabled.<br>1 Associated pin is configured as output.<br>0 Associated pin is configured as input. |
| 5<br>DDRJ | **Port J data direction**—<br>This register controls the data direction of pin 5.<br>The enabled $\overline{CS2}$ signal forces the I/O state to be an output. The enabled IIC1 module forces this pin to be a open drain output. In those cases the data direction bits will not change. The DDRM bits revert to controlling the I/O direction of a pin when the associated peripheral module is disabled.<br>1 Associated pin is configured as output.<br>0 Associated pin is configured as input. |
| 4<br>DDRJ | **Port J data direction**—<br>This register controls the data direction of pin 4.<br>The enabled $\overline{CS0}$ signal forces the I/O state to be an output. The enabled IIC1 module forces this pin to be a open drain output. In those cases the data direction bits will not change. The DDRM bits revert to controlling the I/O direction of a pin when the associated peripheral module is disabled.<br>1 Associated pin is configured as output.<br>0 Associated pin is configured as input. |
| 3<br>DDRJ | **Port J data direction**—<br>This register controls the data direction of pin 3.<br>1 Associated pin is configured as output.<br>0 Associated pin is configured as input. |
| 2<br>DDRJ | **Port J data direction**—<br>This register controls the data direction of pin 2.<br>The enabled $\overline{CS1}$ signal forces the I/O state to be an output. In those cases the data direction bits will not change. The DDRM bits revert to controlling the I/O direction of a pin when the associated peripheral module is disabled.<br>1 Associated pin is configured as output.<br>0 Associated pin is configured as input. |

**Table 2-59. DDRJ Register Field Descriptions (continued)**

| Field | Description |
|---|---|
| 1<br>DDRJ | **Port J data direction**—<br>This register controls the data direction of pin 1.<br>The enabled SCI2 forces the I/O state to be an output. The DDRM bits revert to controlling the I/O direction of a pin when the associated peripheral module is disabled.<br>1 Associated pin is configured as output.<br>0 Associated pin is configured as input. |
| 0<br>DDRJ | **Port J data direction**—<br>This register controls the data direction of pin 0.<br>The enabled SCI3 or $\overline{CS3}$ signal forces the I/O state to be an output. In those cases the data direction bits will not change. The DDRM bits revert to controlling the I/O direction of a pin when the associated peripheral module is disabled.<br>1 Associated pin is configured as output.<br>0 Associated pin is configured as input. |

**NOTE**

Due to internal synchronization circuits, it can take up to 2 bus clock cycles until the correct value is read on PTH or PTIH registers, when changing the DDRH register.

## 2.3.64 Port J Reduced Drive Register (RDRJ)

Address 0x026B — Access: User read/write(1)

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | RDRJ7 | RDRJ6 | RDRJ5 | RDRJ4 | RDRJ3 | RDRJ2 | RDRJ1 | RDRJ0 |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 2-62. Port J Reduced Drive Register (RDRJ)**

1. Read: Anytime.
   Write: Anytime.

**Table 2-60. RDRJ Register Field Descriptions**

| Field | Description |
|---|---|
| 7-0<br>RDRJ | **Port J reduced drive**—Select reduced drive for outputs<br>This register configures the drive strength of output pins 7 through 0 as either full or reduced independent of the function used on the pins. If a pin is used as input this bit has no effect.<br>1 Reduced drive selected (approx. 1/5 of the full drive strength).<br>0 Full drive strength enabled. |

## 2.3.65 Port J Pull Device Enable Register (PERJ)

Address 0x026C Access: User read/write(1)

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | PERJ7 | PERJ6 | PERJ5 | PERJ4 | PERJ3 | PERJ2 | PERJ1 | PERJ0 |
| Reset | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

**Figure 2-63. Port J Pull Device Enable Register (PERJ)**

1. Read: Anytime.
   Write: Anytime.

**Table 2-61. PERJ Register Field Descriptions**

| Field | Description |
|---|---|
| 7-0<br>PERJ | **Port J pull device enable**—Enable pull devices on input pins<br>These bits configure whether a pull device is activated, if the associated pin is used as an input. This bit has no effect if the pin is used as an output. Out of reset all pull device are enabled.<br>1 Pull device enabled.<br>0 Pull device disabled. |

## 2.3.66 Port J Polarity Select Register (PPSJ)

Address 0x026D Access: User read/write(1)

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | PPSJ7 | PPSJ6 | PPSJ5 | PPSJ4 | PPSJ3 | PPSJ2 | PPSJ1 | PPSJ0 |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 2-64. Port J Polarity Select Register (PPSJ)**

1. Read: Anytime.
   Write: Anytime.

**Table 2-62. PPSJ Register Field Descriptions**

| Field | Description |
|---|---|
| 7-0<br>PPSJ | **Port J pull device select**—Determine pull device polarity on input pins<br>This register serves a dual purpose by selecting the polarity of the active interrupt edge as well as selecting a pull-up or pull-down device if enabled.<br>1 A rising edge on the associated Port J pin sets the associated flag bit in the PIFJ register. A pull-down device is connected to the associated Port J pin, if enabled by the associated bit in register PERJ and if the port is used as input.<br>0 A falling edge on the associated Port J pin sets the associated flag bit in the PIFJ register.A pull-up device is connected to the associated Port J pin, if enabled by the associated bit in register PERJ and if the port is used as input. |

## 2.3.67 Port J Interrupt Enable Register (PIEJ)

Address 0x026E — Access: User read/write[1]

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | PIEJ7 | PIEJ6 | PIEJ5 | PIEJ4 | PIEJ3 | PIEJ2 | PIEJ1 | PIEJ0 |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 2-65. Port J Interrupt Enable Register (PIEJ)**

1. Read: Anytime.
   Write: Anytime.

**Table 2-63. PPSP Register Field Descriptions**

| Field | Description |
|---|---|
| 7-0<br>PIEJ | **Port J interrupt enable**—<br>This register disables or enables on a per-pin basis the edge sensitive external interrupt associated with Port J.<br>1 Interrupt is enabled.<br>0 Interrupt is disabled (interrupt flag masked). |

## 2.3.68 Port J Interrupt Flag Register (PIFJ)

Address 0x026F — Access: User read/write[1]

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | PIFJ7 | PIFJ6 | PIFJ5 | PIFJ4 | PIFJ3 | PIFJ2 | PIFJ1 | PIFJ0 |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 2-66. Port J Interrupt Flag Register (PIFJ)**

1. Read: Anytime.
   Write: Anytime.

**Table 2-64. PPSP Register Field Descriptions**

| Field | Description |
|---|---|
| 7-0<br>PIFJ | **Port J interrupt flag**—<br>Each flag is set by an active edge on the associated input pin. This could be a rising or a falling edge based on the state of the PPSJ register. To clear this flag, write logic level 1 to the corresponding bit in the PIFJ register. Writing a 0 has no effect.<br>1 Active edge on the associated bit has occurred (an interrupt will occur if the associated enable bit is set).<br>0 No active edge pending. |

## 2.3.69 Port AD0 Data Register 0 (PT0AD0)

Address 0x0270 Access: User read/write(1)

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | PT0AD07 | PT0AD06 | PT0AD05 | PT0AD04 | PT0AD03 | PT0AD02 | PT0AD01 | PT0AD00 |
| Altern. Function | AN15 | AN14 | AN13 | AN12 | AN11 | AN10 | AN9 | AN8 |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 2-67. Port AD0 Data Register 0 (PT0AD0)**

1. Read: Anytime.
   Write: Anytime.

**Table 2-65. PT0AD0 Register Field Descriptions**

| Field | Description |
|---|---|
| 7-0<br>PT0AD0 | **Port AD0 general purpose input/output data**—Data Register<br>This register is associated with ATD0 analog inputs AN[15:8] on PAD[15:8], respectively.<br>When not used with the alternative function, this pin can be used as general purpose I/O.<br>If the associated data direction bits of these pins are set to 1, a read returns the value of the port register, otherwise the buffered pin input state is read. |

## 2.3.70 Port AD0 Data Register 1 (PT1AD0)

Address 0x0271 Access: User read/write(1)

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | PT1AD07 | PT1AD06 | PT1AD05 | PT1AD04 | PT1AD03 | PT1AD02 | PT1AD01 | PT1AD00 |
| Altern. Function | AN7 | AN6 | AN5 | AN4 | AN3 | AN2 | AN1 | AN0 |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 2-68. Port AD0 Data Register 1 (PT1AD0)**

1. Read: Anytime.
   Write: Anytime.

**Table 2-66. PT1AD0 Register Field Descriptions**

| Field | Description |
|---|---|
| 7-0<br>PT1AD0 | **Port AD0 general purpose input/output data**—Data Register<br>This register is associated with ATD0 analog inputs AN[7:0] on PAD[7:0], respectively.<br>When not used with the alternative function, these pins can be used as general purpose I/O.<br>If the associated data direction bits of these pins are set to 1, a read returns the value of the port register, otherwise the buffered pin input state is read. |

## 2.3.71 Port AD0 Data Direction Register 0 (DDR0AD0)

Address 0x0272 Access: User read/write(1)

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | DDR0AD07 | DDR0AD06 | DDR0AD05 | DDR0AD04 | DDR0AD03 | DDR0AD02 | DDR0AD01 | DDR0AD00 |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 2-69. Port AD0 Data Direction Register 0 (DDR0AD0)**

1. Read: Anytime.
   Write: Anytime.

**Table 2-67. DDR0AD0 Register Field Descriptions**

| Field | Description |
|---|---|
| 7-0<br>DDR0AD0 | **Port AD0 data direction**—<br>This register controls the data direction of pins 15 through 8.<br>1 Associated pin is configured as output.<br>0 Associated pin is configured as input. |

**NOTE**

Due to internal synchronization circuits, it can take up to 2 bus clock cycles until the correct value is read on PT0AD0 registers, when changing the DDR0AD0 register.

**NOTE**

To use the digital input function on Port AD0 the ATD Digital Input Enable Register (ATD0DIEN1) has to be set to logic level “1”.

## 2.3.72 Port AD0 Data Direction Register 1 (DDR1AD0)

Address 0x0273 Access: User read/write(1)

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | DDR1AD07 | DDR1AD06 | DDR1AD05 | DDR1AD04 | DDR1AD03 | DDR1AD02 | DDR1AD01 | DDR1AD00 |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 2-70. Port AD0 Data Direction Register 1 (DDR1AD0)**

1. Read: Anytime.
   Write: Anytime.

**Table 2-68. DDR1AD0 Register Field Descriptions**

| Field | Description |
|---|---|
| 7-0<br>DDR1AD0 | **Port AD0 data direction**—<br>This register controls the data direction of pins 7 through 0.<br>1 Associated pin is configured as output.<br>0 Associated pin is configured as input. |

**NOTE**

Due to internal synchronization circuits, it can take up to 2 bus clock cycles until the correct value is read on PT0AD0 registers, when changing the DDR1AD0 register.

**NOTE**

To use the digital input function on Port AD0 the ATD Digital Input Enable Register (ATD0DIEN1) has to be set to logic level “1”.

## 2.3.73 Port AD0 Reduced Drive Register 0 (RDR0AD0)

Address 0x0274 Access: User read/write(1)

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | RDR0AD07 | RDR0AD06 | RDR0AD05 | RDR0AD04 | RDR0AD03 | RDR0AD02 | RDR0AD01 | RDR0AD00 |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 2-71. Port AD0 Reduced Drive Register 0 (RDR0AD0)**

1. Read: Anytime.
   Write: Anytime.

**Table 2-69. RDR0AD0 Register Field Descriptions**

| Field | Description |
|---|---|
| 7-0<br>RDR0AD0 | **Port AD0 reduced drive**—Select reduced drive for Port AD0 outputs<br>This register configures the drive strength of Port AD0 output pins 15 through 8 as either full or reduced independent of the function used on the pins. If a pin is used as input this bit has no effect.<br>1 Reduced drive selected (approx. 1/5 of the full drive strength).<br>0 Full drive strength enabled. |

## 2.3.74 Port AD0 Reduced Drive Register 1 (RDR1AD0)

Address 0x0275 Access: User read/write(1)

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | RDR1AD07 | RDR1AD06 | RDR1AD05 | RDR1AD04 | RDR1AD03 | RDR1AD02 | RDR1AD01 | RDR1AD00 |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 2-72. Port AD0 Reduced Drive Register 1 (RDR1AD0)**

1. Read: Anytime.
   Write: Anytime.

**Table 2-70. RDR1AD0 Register Field Descriptions**

| Field | Description |
|---|---|
| 7-0<br>RDR1AD0 | **Port AD0 reduced drive**—Select reduced drive for Port AD0 outputs<br>This register configures the drive strength of Port AD0 output pins 7 through 0 as either full or reduced independent of the function used on the pins. If a pin is used as input this bit has no effect.<br>1 Reduced drive selected (approx. 1/5 of the full drive strength).<br>0 Full drive strength enabled. |

## 2.3.75 Port AD0 Pull Up Enable Register 0 (PER0AD0)

Address 0x0276 Access: User read/write(1)

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | PER0AD07 | PER0AD06 | PER0AD05 | PER0AD04 | PER0AD03 | PER0AD02 | PER0AD01 | PER0AD00 |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 2-73. Port AD0 Pull Device Up Register 0 (PER0AD0)**

1. Read: Anytime.
   Write: Anytime.

**Table 2-71. PER0AD0 Register Field Descriptions**

| Field | Description |
|---|---|
| 7-0<br>PER0AD0 | **Port AD0 pull device enable**—Enable pull devices on input pins<br>These bits configure whether a pull device is activated, if the associated pin is used as an input. This bit has no effect if the pin is used as an output. Out of reset no pull device is enabled.<br>1 Pull device enabled.<br>0 Pull device disabled. |

## 2.3.76 Port AD0 Pull Up Enable Register 1 (PER1AD0)

Address 0x0277 Access: User read/write(1)

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | PER1AD07 | PER1AD06 | PER1AD05 | PER1AD04 | PER1AD03 | PER1AD02 | PER1AD01 | PER1AD00 |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 2-74. Port AD0 Pull Up Enable Register 1 (PER1AD0)**

1. Read: Anytime.
   Write: Anytime.

**Table 2-72. PER1AD0 Register Field Descriptions**

| Field | Description |
|---|---|
| 7-0<br>PER1AD0 | **Port AD0 pull device enable**—Enable pull devices on input pins<br>These bits configure whether a pull device is activated, if the associated pin is used as an input. This bit has no effect if the pin is used as an output. Out of reset no pull device is enabled.<br>1 Pull device enabled.<br>0 Pull device disabled. |

## 2.3.77 Port AD1 Data Register 0 (PT0AD1)

Address 0x0278 Access: User read/write(1)

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | PT0AD17 | PT0AD16 | PT0AD15 | PT0AD14 | PT0AD13 | PT0AD12 | PT0AD11 | PT0AD10 |
| Altern.<br>Function | AN15 | AN14 | AN13 | AN12 | AN11 | AN10 | AN9 | AN8 |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 2-75. Port AD1 Data Register 0 (PT0AD1)**

1. Read: Anytime.
   Write: Anytime.

**Table 2-73. PT0AD1 Register Field Descriptions**

| Field | Description |
|---|---|
| 7-0<br>PT0AD1 | **Port AD1 general purpose input/output data**—Data Register<br>This register is associated with ATD1 analog inputs AN[15:8] on PAD[31:24], respectively.<br>When not used with the alternative function, this pin can be used as general purpose I/O.<br>If the associated data direction bits of these pins are set to 1, a read returns the value of the port register, otherwise the buffered pin input state is read. |

## 2.3.78 Port AD1 Data Register 1 (PT1AD1)

Address 0x0279 Access: User read/write(1)

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | PT1AD17 | PT1AD16 | PT1AD15 | PT1AD14 | PT1AD13 | PT1AD12 | PT1AD11 | PT1AD10 |
| Altern. Function | AN7 | AN6 | AN5 | AN4 | AN3 | AN2 | AN1 | AN0 |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 2-76. Port AD1 Data Register 1 (PT1AD1)**

1. Read: Anytime.
   Write: Anytime.

**Table 2-74. PT1AD1 Register Field Descriptions**

| Field | Description |
|---|---|
| 7-0<br>PT1AD1 | **Port AD1 general purpose input/output data**—Data Register<br>This register is associated with ATD1 analog inputs AN[7:0] on PAD[23:16], respectively.<br>When not used with the alternative function, these pins can be used as general purpose I/O.<br>If the associated data direction bits of these pins are set to 1, a read returns the value of the port register, otherwise the buffered pin input state is read. |

## 2.3.79 Port AD1 Data Direction Register 0 (DDR0AD1)

Address 0x027A Access: User read/write(1)

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | DDR0AD17 | DDR0AD16 | DDR0AD15 | DDR0AD14 | DDR0AD13 | DDR0AD12 | DDR0AD11 | DDR0AD10 |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 2-77. Port AD1 Data Direction Register 0 (DDR0AD1)**

1. Read: Anytime.
   Write: Anytime.

**Table 2-75. DDR0AD1 Register Field Descriptions**

| Field | Description |
|---|---|
| 7-0<br>DDR0AD1 | **Port AD1 data direction**—<br>This register controls the data direction of pins 15 through 8.<br>1 Associated pin is configured as output.<br>0 Associated pin is configured as input. |

**NOTE**

Due to internal synchronization circuits, it can take up to 2 bus clock cycles until the correct value is read on PT0AD1 registers, when changing the DDR0AD1 register.

**NOTE**

To use the digital input function on Port AD1 the ATD Digital Input Enable Register (ATD1DIEN1) has to be set to logic level “1”.

## 2.3.80 Port AD1 Data Direction Register 1 (DDR1AD1)

Address 0x027B — Access: User read/write(1)

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | DDR1AD17 | DDR1AD16 | DDR1AD15 | DDR1AD14 | DDR1AD13 | DDR1AD12 | DDR1AD11 | DDR1AD10 |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 2-78. Port AD1 Data Direction Register 1 (DDR1AD1)**

1. Read: Anytime.
   Write: Anytime.

**Table 2-76. DDR1AD1 Register Field Descriptions**

| Field | Description |
|---|---|
| 7-0<br>DDR1AD1 | **Port AD1 data direction**—<br>This register controls the data direction of pins 7 through 0.<br>1 Associated pin is configured as output.<br>0 Associated pin is configured as input. |

**NOTE**

Due to internal synchronization circuits, it can take up to 2 bus clock cycles until the correct value is read on PT0AD1 registers, when changing the DDR1AD1 register.

**NOTE**

To use the digital input function on Port AD1 the ATD Digital Input Enable Register (ATD1DIEN1) has to be set to logic level “1”.

## 2.3.81 Port AD1 Reduced Drive Register 0 (RDR0AD1)

Address 0x027C Access: User read/write(1)

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | RDR0AD17 | RDR0AD16 | RDR0AD15 | RDR0AD14 | RDR0AD13 | RDR0AD12 | RDR0AD11 | RDR0AD10 |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 2-79. Port AD1 Reduced Drive Register 0 (RDR0AD1)**

1. Read: Anytime.
   Write: Anytime.

**Table 2-77. RDR0AD1 Register Field Descriptions**

| Field | Description |
|---|---|
| 7-0<br>RDR0AD1 | **Port AD1 reduced drive**—Select reduced drive for Port AD1 outputs<br>This register configures the drive strength of Port AD1 output pins 15 through 8 as either full or reduced independent of the function used on the pins. If a pin is used as input this bit has no effect.<br>1 Reduced drive selected (approx. 1/5 of the full drive strength).<br>0 Full drive strength enabled. |

## 2.3.82 Port AD1 Reduced Drive Register 1 (RDR1AD1)

Address 0x027D Access: User read/write(1)

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | RDR1AD17 | RDR1AD16 | RDR1AD15 | RDR1AD14 | RDR1AD13 | RDR1AD12 | RDR1AD11 | RDR1AD10 |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 2-80. Port AD1 Reduced Drive Register 1 (RDR1AD1)**

1. Read: Anytime.
   Write: Anytime.

**Table 2-78. RDR1AD1 Register Field Descriptions**

| Field | Description |
|---|---|
| 7-0<br>RDR1AD1 | **Port AD1 reduced drive**—Select reduced drive for Port AD1 outputs<br>This register configures the drive strength of Port AD1 output pins 7 through 0 as either full or reduced independent of the function used on the pins. If a pin is used as input this bit has no effect.<br>1 Reduced drive selected (approx. 1/5 of the full drive strength).<br>0 Full drive strength enabled. |

## 2.3.83 Port AD1 Pull Up Enable Register 0 (PER0AD1)

Address 0x027E Access: User read/write(1)

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | PER0AD17 | PER0AD16 | PER0AD15 | PER0AD14 | PER0AD13 | PER0AD12 | PER0AD11 | PER0AD10 |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 2-81. Port AD1 Pull Device Up Register 0 (PER0AD1)**

1. Read: Anytime.
   Write: Anytime.

**Table 2-79. PER0AD1 Register Field Descriptions**

| Field | Description |
|---|---|
| 7-0<br>PER0AD1 | **Port AD1 pull device enable**—Enable pull devices on input pins<br>These bits configure whether a pull device is activated, if the associated pin is used as an input. This bit has no effect if the pin is used as an output. Out of reset no pull device is enabled.<br>1 Pull device enabled.<br>0 Pull device disabled. |

## 2.3.84 Port AD1 Pull Up Enable Register 1 (PER1AD1)

Address 0x027F Access: User read/write(1)

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | PER1AD17 | PER1AD16 | PER1AD15 | PER1AD14 | PER1AD13 | PER1AD12 | PER1AD11 | PER1AD10 |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 2-82. Port AD1 Pull Up Enable Register 1 (PER1AD1)**

1. Read: Anytime.
   Write: Anytime.

**Table 2-80. PER1AD1 Register Field Descriptions**

| Field | Description |
|---|---|
| 7-0<br>PER1AD1 | **Port AD1 pull device enable**—Enable pull devices on input pins<br>These bits configure whether a pull device is activated, if the associated pin is used as an input. This bit has no effect if the pin is used as an output. Out of reset no pull device is enabled.<br>1 Pull device enabled.<br>0 Pull device disabled. |

## 2.3.85 Port R Data Register (PTR)

Address 0x0368 Access: User read/write(1)

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | PTR7 | PTR6 | PTR5 | PTR4 | PTR3 | PTR2 | PTR1 | PTR0 |
| Altern. Function | TIMIOC7 | TIMIOC6 | TIMIOC5 | TIMIOC4 | TIMIOC3 | TIMIOC2 | TIMIOC1 | TIMIOC0 |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 2-83. Port R Data Register (PTR)**

1. Read: Anytime.
   Write: Anytime.

**Table 2-81. PTR Register Field Descriptions**

| Field | Description |
|---|---|
| 7-0<br>PTR | **Port R general purpose input/output data**—Data Register<br>Port R pins 7 through 0 are associated with TIM channels TIMIOC7 through TIMIOC0.<br>When not used with the alternative function, these pins can be used as general purpose I/O.<br>If the associated data direction bit of this pin is set to 1, a read returns the value of the port register, otherwise the buffered pin input state is read. |

## 2.3.86 Port R Input Register (PTIR)

Address 0x0369 Access: User read(1)

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | PTIR7 | PTIR6 | PTIR5 | PTIR4 | PTIR3 | PTIR2 | PTIR1 | PTIR0 |
| W | | | | | | | | |
| Reset | u | u | u | u | u | u | u | u |

= Unimplemented or Reserved u = Unaffected by reset

**Figure 2-84. Port R Input Register (PTIR)**

1. Read: Anytime.
   Write:Never, writes to this register have no effect.

**Table 2-82. PTIR Register Field Descriptions**

| Field | Description |
|---|---|
| 7-0<br>PTIR | **Port R input data**—<br>This register always reads back the buffered state of the associated pins. This can also be used to detect overload or short circuit conditions on output pins. |

## 2.3.87 Port R Data Direction Register (DDRR)

Address 0x036A Access: User read/write(1)

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | DDRR7 | DDRR6 | DDRR5 | DDRR4 | DDRR3 | DDRR2 | DDRR1 | DDRR0 |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 2-85. Port R Data Direction Register (DDRR)**

1. Read: Anytime.
   Write: Anytime.

**Table 2-83. DDRR Register Field Descriptions**

| Field | Description |
|---|---|
| 7-0<br>DDRR | **Port R data direction**—<br>This register controls the data direction of pins 7 through 0.<br>The TIM forces the I/O state to be an output for each timer port associated with an enabled output compare. In this case the data direction bits will not change.<br>The data direction bits revert to controlling the I/O direction of a pin when the associated timer output compare is disabled.<br>The timer Input Capture always monitors the state of the pin.<br>1 Associated pin is configured as output.<br>0 Associated pin is configured as high-impedance input. |

**NOTE**

Due to internal synchronization circuits, it can take up to 2 bus clock cycles until the correct value is read on PTR or PTIR registers, when changing the DDRR register.

## 2.3.88 Port R Reduced Drive Register (RDRR)

Address 0x036B Access: User read/write(1)

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | RDRR7 | RDRR6 | RDRR5 | RDRR4 | RDRR3 | RDRR2 | RDRR1 | RDRR0 |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 2-86. Port R Reduced Drive Register (RDRR)**

1. Read: Anytime.
   Write: Anytime.

**Table 2-84. RDRR Register Field Descriptions**

| Field | Description |
|---|---|
| 7-0<br>RDRR | **Port R reduced drive**—Select reduced drive for outputs<br>This register configures the drive strength of output pins 7 through 0 as either full or reduced independent of the function used on the pins. If a pin is used as input this bit has no effect.<br>1 Reduced drive selected (approx. 1/5 of the full drive strength).<br>0 Full drive strength enabled. |

## 2.3.89 Port R Pull Device Enable Register (PERR)

Address 0x036C Access: User read/write(1)

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | PERR7 | PERR6 | PERR5 | PERR4 | PERR3 | PERR2 | PERR1 | PERR0 |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 2-87. Port R Pull Device Enable Register (PERR)**

1. Read: Anytime.
   Write: Anytime.

**Table 2-85. PERR Register Field Descriptions**

| Field | Description |
|---|---|
| 7-0<br>PERR | **Port R pull device enable**—Enable pull devices on input pins<br>These bits configure whether a pull device is activated, if the associated pin is used as an input. This bit has no effect if the pin is used as an output. Out of reset no pull device is enabled.<br>1 Pull device enabled.<br>0 Pull device disabled. |

## 2.3.90 Port R Polarity Select Register (PPSR)

Address 0x036D Access: User read/write(1)

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | PPSR7 | PPSR6 | PPSR5 | PPSR4 | PPSR3 | PPSR2 | PPSR1 | PPSR0 |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 2-88. Port R Polarity Select Register (PPSR)**

1. Read: Anytime.
   Write: Anytime.

**Table 2-86. PPSR Register Field Descriptions**

| Field | Description |
|---|---|
| 7-0<br>PPSR | **Port R pull device select**—Determine pull device polarity on input pins<br>This register selects whether a pull-down or a pull-up device is connected to the pin.<br>1 A pull-down device is connected to the associated pin, if enabled and if the pin is used as input.<br>0 A pull-up device is connected to the associated pin, if enabled and if the pin is used as input. |

## 2.3.91 PIM Reserved Register

Address 0x036E Access: User read(1)

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| W | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented or Reserved

**Figure 2-89. PIM Reserved Register**

1. Read: Always reads 0x00
   Write: Unimplemented

## 2.3.92 Port R Routing Register (PTRRR)

Address 0x036F Access: User read/write(1)

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | PTRRR7 | PTRRR6 | PTRRR5 | PTRRR4 | PTRRR3 | PTRRR2 | PTRRR1 | PTRRR0 |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented or Reserved

**Figure 2-90. Port R Routing Register (PTRRR)**

1. Read: Anytime.
   Write: Anytime.

**Table 2-87. PTR Routing Register Field Descriptions**

| Field | Description |
|---|---|
| 7<br>PTRRR | **Port R routing**—<br>This register configures the re-routing of the associated TIM channel.<br>1 TIMIOC7 is available on PP7<br>0 TIMIOC7 is available on PR7 |
| 6<br>PTRRR | **Port R routing**—<br>This register configures the re-routing of the associated TIM channel.<br>1 TIMIOC6 is available on PP6<br>0 TIMIOC6 is available on PR6 |

**Table 2-87. PTR Routing Register Field Descriptions (continued)**

| Field | Description |
|---|---|
| 5<br>PTRRR | **Port R routing**—<br>This register configures the re-routing of the associated TIM channel.<br>1 TIMIOC5 is available on PP5<br>0 TIMIOC5 is available on PR5 |
| 4<br>PTRRR | **Port R routing**—<br>This register configures the re-routing of the associated TIM channel.<br>1 TIMIOC4 is available on PP4<br>0 TIMIOC4 is available on PR4 |
| 3<br>PTRRR | **Port R routing**—<br>This register configures the re-routing of the associated TIM channel.<br>1 TIMIOC3 is available on PP3<br>0 TIMIOC3 is available on PR3 |
| 2<br>PTRRR | **Port R routing**—<br>This register configures the re-routing of the associated TIM channel.<br>1 TIMIOC2 is available on PP2<br>0 TIMIOC2 is available on PR2 |
| 1<br>PTRRR | **Port R routing**—<br>This register configures the re-routing of the associated TIM channel.<br>1 TIMIOC1 is available on PP1<br>0 TIMIOC1 is available on PR1 |
| 0<br>PTRRR | **Port R routing**—<br>This register configures the re-routing of the associated TIM channel.<br>1 TIMIOC0 is available on PP0<br>0 TIMIOC0 is available on PR0 |

## 2.3.93 Port L Data Register (PTL)

Address 0x0370 Access: User read/write(1)

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | PTL7 | PTLT6 | PTL5 | PTL4 | PTL3 | PTL2 | PTL1 | PTL0 |
| Altern. Function | (TXD7) | (RXD7) | (TXD6) | (RXD6) | (TXD5) | (RXD5) | (TXD4) | (RXD4) |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 2-91. Port L Data Register (PTL)**

1. Read: Anytime.
   Write: Anytime.

**Table 2-88. PTL Register Field Descriptions**

| Field | Description |
|---|---|
| 7<br>PTL | **Port L general purpose input/output data**—Data Register<br>Port L pin 7 is associated with the TXD signal of the SCI7 module.<br>When not used with the alternative function, this pin can be used as general purpose I/O.<br>If the associated data direction bit of this pin is set to 1, a read returns the value of the port register, otherwise the buffered pin input state is read. |
| 6<br>PTL | **Port L general purpose input/output data**—Data Register<br>Port L pin 6 is associated with the RXD signal of the SCI7 module.<br>When not used with the alternative function, this pin can be used as general purpose I/O.<br>If the associated data direction bit of this pin is set to 1, a read returns the value of the port register, otherwise the buffered pin input state is read. |
| 5<br>PTL | **Port L general purpose input/output data**—Data Register<br>Port L pin 5 is associated with the TXD signal of the SCI6 module.<br>When not used with the alternative function, this pin can be used as general purpose I/O.<br>If the associated data direction bit of this pin is set to 1, a read returns the value of the port register, otherwise the buffered pin input state is read. |
| 4<br>PTL | **Port L general purpose input/output data**—Data Register<br>Port L pin 4 is associated with the RXD signal of the SCI6 module.<br>When not used with the alternative function, this pin can be used as general purpose I/O.<br>If the associated data direction bit of this pin is set to 1, a read returns the value of the port register, otherwise the buffered pin input state is read. |
| 3<br>PTL | **Port L general purpose input/output data**—Data Register<br>Port L pin 3 is associated with the TXD signal of the SC5 module.<br>When not used with the alternative function, this pin can be used as general purpose I/O.<br>If the associated data direction bit of this pin is set to 1, a read returns the value of the port register, otherwise the buffered pin input state is read. |
| 2<br>PTL | **Port L general purpose input/output data**—Data Register<br>Port L pin 2 is associated with the RXD signal of the SCI5 module.<br>When not used with the alternative function, this pin can be used as general purpose I/O.<br>If the associated data direction bit of this pin is set to 1, a read returns the value of the port register, otherwise the buffered pin input state is read. |
| 1<br>PTL | **Port L general purpose input/output data**—Data Register<br>Port L pin 3 is associated with the TXD signal of the SCI4 module.<br>When not used with the alternative function, this pin can be used as general purpose I/O.<br>If the associated data direction bit of this pin is set to 1, a read returns the value of the port register, otherwise the buffered pin input state is read. |
| 0<br>PTL | **Port L general purpose input/output data**—Data Register<br>Port L pin 2 is associated with the RXD signal of the SCI4 module.<br>When not used with the alternative function, this pin can be used as general purpose I/O.<br>If the associated data direction bit of this pin is set to 1, a read returns the value of the port register, otherwise the buffered pin input state is read. |

## 2.3.94 Port L Input Register (PTIL)

Address 0x0371 Access: User read(1)

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | PTIL7 | PTIL6 | PTIL5 | PTIL4 | PTIL3 | PTIL2 | PTIL1 | PTIL0 |
| W | | | | | | | | |
| Reset | u | u | u | u | u | u | u | u |

= Unimplemented or Reserved u = Unaffected by reset

**Figure 2-92. Port L Input Register (PTIL)**

1. Read: Anytime.
   Write:Never, writes to this register have no effect.

**Table 2-89. PTIL Register Field Descriptions**

| Field | Description |
|---|---|
| 7-0<br>PTIL | **Port L input data**—<br>This register always reads back the buffered state of the associated pins. This can also be used to detect overload or short circuit conditions on output pins. |

## 2.3.95 Port L Data Direction Register (DDRL)

Address 0x0372 Access: User read/write(1)

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | DDRL7 | DDRL6 | DDRL5 | DDRL4 | DDRL3 | DDRL2 | DDRL1 | DDRL0 |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 2-93. Port L Data Direction Register (DDRL)**

1. Read: Anytime.
   Write: Anytime.

**Table 2-90. DDRL Register Field Descriptions**

| Field | Description |
|---|---|
| 7-0<br>DDRL | **Port L data direction**—<br>This register controls the data direction of pins 7 through 0.This register configures each Port L pin as either input or output.<br>If SPI0 is enabled, the SPI0 determines the pin direction. *Refer to SPI section for details*.<br>If the associated SCI transmit or receive channel is enabled this register has no effect on the pins. The pin is forced to be an output if a SCI transmit channel is enabled, it is forced to be an input if the SCI receive channel is enabled. The data direction bits revert to controlling the I/O direction of a pin when the associated channel is disabled.<br>1 Associated pin is configured as output.<br>0 Associated pin is configured as input. |

**NOTE**

Due to internal synchronization circuits, it can take up to 2 bus clock cycles until the correct value is read on PTL or PTIL registers, when changing the DDRL register.

## 2.3.96 Port L Reduced Drive Register (RDRL)

Address 0x0373 Access: User read/write(1)

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | RDRL7 | RDRL6 | RDRL5 | RDRL4 | RDRL3 | RDRL2 | RDRL1 | RDRL0 |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 2-94. Port L Reduced Drive Register (RDRL)**

1. Read: Anytime.
   Write: Anytime.

**Table 2-91. RDRL Register Field Descriptions**

| Field | Description |
|---|---|
| 7-0<br>RDRL | **Port L reduced drive**—Select reduced drive for outputs<br>This register configures the drive strength of output pins 7 through 0 as either full or reduced independent of the function used on the pins. If a pin is used as input this bit has no effect.<br>1 Reduced drive selected (approx. 1/5 of the full drive strength).<br>0 Full drive strength enabled. |

## 2.3.97 Port L Pull Device Enable Register (PERL)

Address 0x0374 Access: User read/write(1)

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | PERL7 | PERL6 | PERL5 | PERL4 | PERL3 | PERL2 | PERL1 | PERL0 |
| Reset | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

**Figure 2-95. Port L Pull Device Enable Register (PERL)**

1. Read: Anytime.
   Write: Anytime.

**Table 2-92. PERL Register Field Descriptions**

| Field | Description |
|---|---|
| 7-0<br>PERL | **Port L pull device enable**—Enable pull devices on input pins<br>These bits configure whether a pull device is activated, if the associated pin is used as an input. This bit has no effect if the pin is used as an output. Out of reset all pull devices are enabled.<br>1 Pull device enabled.<br>0 Pull device disabled. |

## 2.3.98 Port L Polarity Select Register (PPSL)

Address 0x0375 Access: User read/write(1)

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | PPSL7 | PPSL6 | PPSL5 | PPSL4 | PPSL3 | PPSL2 | PPSL1 | PPSL0 |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 2-96. Port L Polarity Select Register (PPSL)**

1. Read: Anytime.
   Write: Anytime.

**Table 2-93. PPSL Register Field Descriptions**

| Field | Description |
|---|---|
| 7-0<br>PPSL | **Port L pull device select**—Determine pull device polarity on input pins<br>This register selects whether a pull-down or a pull-up device is connected to the pin.<br>1 A pull-down device is connected to the associated pin, if enabled and if the pin is used as input.<br>0 A pull-up device is connected to the associated pin, if enabled and if the pin is used as input. |

## 2.3.99 Port L Wired-Or Mode Register (WOML)

Address 0x0376 Access: User read/write(1)

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | WOML7 | WOML6 | WOML5 | WOML4 | WOML3 | WOML2 | WOML1 | WOML0 |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 2-97. Port L Wired-Or Mode Register (WOML)**

1. Read: Anytime.
   Write: Anytime.

**Table 2-94. WOML Register Field Descriptions**

| Field | Description |
|---|---|
| 7-0<br>WOML | **Port L wired-or mode**—Enable wired-or functionality<br>This register configures the output pins as wired-or independent of the function used on the pins. If enabled the output is driven active low only (open-drain). A logic level of “1” is not driven.This allows a multipoint connection of several serial modules. These bits have no influence on pins used as inputs.<br>1 Output buffers operate as open-drain outputs.<br>0 Output buffers operate as push-pull outputs. |

## 2.3.100 Port L Routing Register (PTLRR)

Address 0x0377 Access: User read/write(1)

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | PTLRR7 | PTLRR6 | PTLRR5 | PTLRR4 | 0 | 0 | 0 | 0 |
| W | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented or Reserved

**Figure 2-98. Port L Routing Register (PTLRR)**

1. Read: Anytime.
   Write: Anytime.

This register configures the re-routing of SCI7, SCI6, SCI5, and SCI4 on alternative ports.

**Table 2-95. Port L Routing Summary**

| Module | PTLRR | | | | Related Pins | |
|---|---|---|---|---|---|---|
| | 7 | 6 | 5 | 4 | | |
| | | | | | TXD | RXD |
| SCI7 | 0 | x | x | x | PH3 | PH2 |
| | 1 | x | x | x | PL7 | PL6 |
| SCI6 | x | 0 | x | x | PH1 | PH0 |
| | x | 1 | x | x | PL5 | PL4 |
| SCI5 | x | x | 0 | x | PH7 | PH6 |
| | x | x | 1 | x | PL3 | PL2 |
| SCI4 | x | x | x | 0 | PH5 | PH4 |
| | x | x | x | 1 | PL1 | PL0 |

## 2.3.101 Port F Data Register (PTF)

Address 0x0378 Access: User read/write(1)

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | PTF7 | PTFT6 | PTF5 | PTF4 | PTF3 | PTF2 | PTF1 | PTF0 |
| W | | | | | | | | |
| Altern. Function | (TXD3) | (RXD3) | (SCL0) | (SDA0) | ($\overline{CS3}$) | ($\overline{CS2}$) | ($\overline{CS1}$) | ($\overline{CS0}$) |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 2-99. Port F Data Register (PTF)**

1. Read: Anytime.
   Write: Anytime.

**Table 2-96. PTF Register Field Descriptions**

| Field | Description |
|---|---|
| 7<br>PTF | **Port F general purpose input/output data**—Data Register<br>Port F pin 7 is associated with the TXD signal of the SCI3 module.<br>When not used with the alternative function, this pin can be used as general purpose I/O.<br>If the associated data direction bit of this pin is set to 1, a read returns the value of the port register, otherwise the buffered pin input state is read. |
| 6<br>PTF | **Port F general purpose input/output data**—Data Register<br>Port F pin 6 is associated with the RXD signal of the SCI3 module.<br>When not used with the alternative function, this pin can be used as general purpose I/O.<br>If the associated data direction bit of this pin is set to 1, a read returns the value of the port register, otherwise the buffered pin input state is read. |
| 5<br>PTF | **Port F general purpose input/output data**—Data Register<br>Port F pin 5 is associated with the TXD signal of the SCI6 module.<br>When not used with the alternative function, this pin can be used as general purpose I/O.<br>If the associated data direction bit of this pin is set to 1, a read returns the value of the port register, otherwise the buffered pin input state is read. |
| 4<br>PTF | **Port F general purpose input/output data**—Data Register<br>Port F pin 4 is associated with the RXD signal of the SCI6 module.<br>When not used with the alternative function, this pin can be used as general purpose I/O.<br>If the associated data direction bit of this pin is set to 1, a read returns the value of the port register, otherwise the buffered pin input state is read. |
| 3<br>PTF | **Port F general purpose input/output data**—Data Register<br>Port F pin 3 is associated with the TXD signal of the SC5 module.<br>When not used with the alternative function, this pin can be used as general purpose I/O.<br>If the associated data direction bit of this pin is set to 1, a read returns the value of the port register, otherwise the buffered pin input state is read. |
| 2<br>PTF | **Port F general purpose input/output data**—Data Register<br>Port F pin 2 is associated with the RXD signal of the SCI5 module.<br>When not used with the alternative function, this pin can be used as general purpose I/O.<br>If the associated data direction bit of this pin is set to 1, a read returns the value of the port register, otherwise the buffered pin input state is read. |
| 1<br>PTF | **Port F general purpose input/output data**—Data Register<br>Port F pin 3 is associated with the TXD signal of the SCI4 module.<br>When not used with the alternative function, this pin can be used as general purpose I/O.<br>If the associated data direction bit of this pin is set to 1, a read returns the value of the port register, otherwise the buffered pin input state is read. |
| 0<br>PTF | **Port F general purpose input/output data**—Data Register<br>Port F pin 2 is associated with the RXD signal of the SCI4 module.<br>When not used with the alternative function, this pin can be used as general purpose I/O.<br>If the associated data direction bit of this pin is set to 1, a read returns the value of the port register, otherwise the buffered pin input state is read. |

## 2.3.102 Port F Input Register (PTIF)

Address 0x0379 Access: User read(1)

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | PTIF7 | PTIF6 | PTIF5 | PTIF4 | PTIF3 | PTIF2 | PTIF1 | PTIF0 |
| W | | | | | | | | |
| Reset | u | u | u | u | u | u | u | u |

= Unimplemented or Reserved u = Unaffected by reset

**Figure 2-100. Port F Input Register (PTIF)**

1. Read: Anytime.
   Write:Never, writes to this register have no effect.

**Table 2-97. PTIF Register Field Descriptions**

| Field | Description |
|---|---|
| 7-0<br>PTIF | **Port F input data**—<br>This register always reads back the buffered state of the associated pins. This can also be used to detect overload or short circuit conditions on output pins. |

## 2.3.103 Port F Data Direction Register (DDRF)

Address 0x037A Access: User read/write(1)

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | DDRF7 | DDRF6 | DDRF5 | DDRF4 | DDRF3 | DDRF2 | DDRF1 | DDRF0 |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 2-101. Port F Data Direction Register (DDRF)**

1. Read: Anytime.
   Write: Anytime.

**Table 2-98. DDRF Register Field Descriptions**

| Field | Description |
|---|---|
| 7-0<br>DDRF | **Port F data direction**—<br>This register controls the data direction of pins 7 through 0.This register configures each Port F pin as either input or output.<br>If SPI0 is enabled, the SPI0 determines the pin direction. *Refer to SPI section for details*.<br>If the associated SCI transmit or receive channel is enabled this register has no effect on the pins. The pin is forced to be an output if a SCI transmit channel is enabled, it is forced to be an input if the SCI receive channel is enabled. The data direction bits revert to controlling the I/O direction of a pin when the associated channel is disabled.<br>1 Associated pin is configured as output.<br>0 Associated pin is configured as input. |

**NOTE**

Due to internal synchronization circuits, it can take up to 2 bus clock cycles until the correct value is read on PTF or PTIF registers, when changing the DDRF register.

## 2.3.104 Port F Reduced Drive Register (RDRF)

Address 0x037B — Access: User read/write(1)

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | RDRF7 | RDRF6 | RDRF5 | RDRF4 | RDRF3 | RDRF2 | RDRF1 | RDRF0 |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 2-102. Port F Reduced Drive Register (RDRF)**

1. Read: Anytime.
   Write: Anytime.

**Table 2-99. RDRF Register Field Descriptions**

| Field | Description |
|---|---|
| 7-0<br>RDRF | **Port F reduced drive**—Select reduced drive for outputs<br>This register configures the drive strength of output pins 7 through 0 as either full or reduced independent of the function used on the pins. If a pin is used as input this bit has no effect.<br>1 Reduced drive selected (approx. 1/5 of the full drive strength).<br>0 Full drive strength enabled. |

## 2.3.105 Port F Pull Device Enable Register (PERF)

Address 0x037C — Access: User read/write(1)

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | PERF7 | PERF6 | PERF5 | PERF4 | PERF3 | PERF2 | PERF1 | PERF0 |
| Reset | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

**Figure 2-103. Port F Pull Device Enable Register (PERF)**

1. Read: Anytime.
   Write: Anytime.

**Table 2-100. PERF Register Field Descriptions**

| Field | Description |
|---|---|
| 7-0<br>PERF | **Port F pull device enable**—Enable pull devices on input pins<br>These bits configure whether a pull device is activated, if the associated pin is used as an input. This bit has no effect if the pin is used as an output. Out of reset all pull devices are enabled.<br>1 Pull device enabled.<br>0 Pull device disabled. |

## 2.3.106 Port F Polarity Select Register (PPSF)

Address 0x037D Access: User read/write(1)

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | PPSF7 | PPSF6 | PPSF5 | PPSF4 | PPSF3 | PPSF2 | PPSF1 | PPSF0 |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 2-104. Port F Polarity Select Register (PPSF)**

1. Read: Anytime.
   Write: Anytime.

**Table 2-101. PPSF Register Field Descriptions**

| Field | Description |
|---|---|
| 7-0<br>PPSF | **Port F pull device select**—Determine pull device polarity on input pins<br>This register selects whether a pull-down or a pull-up device is connected to the pin.<br>1 A pull-down device is connected to the associated pin, if enabled and if the pin is used as input.<br>0 A pull-up device is connected to the associated pin, if enabled and if the pin is used as input. |

## 2.3.107 PIM Reserved Register

Address 0x037E Access: User read(1)

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| W | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented or Reserved

**Figure 2-105. PIM Reserved Register**

1. Read: Always reads 0x00
   Write: Unimplemented

## 2.3.108 Port F Routing Register (PTFRR)

Address 0x037F Access: User read/write(1)

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | PTFRR5 | PTFRR4 | PTFRR3 | PTFRR2 | PTFRR1 | PTFRR0 |
| W | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented or Reserved

**Figure 2-106. Port F Routing Register (PTFRR)**

1. Read: Anytime.
   Write: Anytime.

This register configures the re-routing of SCI3, IIC0, $\overline{CS}$[3:0] on alternative ports.

**Table 2-102. Port F Routing Summary**

| Module | PTFRR | | | | | | Related Pins | |
|---|---|---|---|---|---|---|---|---|
| | 5 | 4 | 3 | 2 | 1 | 0 | | |
| | | | | | | | **TXD** | **RXD** |
| SCI3 | 0 | x | x | x | x | x | PM7 | PM6 |
| | 1 | x | x | x | x | x | PF7 | PF6 |
| | | | | | | | **SCL** | **SDA** |
| IIC0 | x | 0 | x | x | x | x | PJ7 | PJ6 |
| | x | 1 | x | x | x | x | PF5 | PF4 |
| | | | | | | | **$\overline{CS}$** | |
| $\overline{CS3}$ | x | x | 0 | x | x | x | PJ0 | |
| | x | x | 1 | x | x | x | PF3 | |
| $\overline{CS2}$ | x | x | x | 0 | x | x | PJ5 | |
| | x | x | x | 1 | x | x | PF2 | |
| $\overline{CS1}$ | x | x | x | x | 0 | x | PJ2 | |
| | x | x | x | x | 1 | x | PF1 | |
| $\overline{CS0}$ | x | x | x | x | x | 0 | PJ4 | |
| | x | x | x | x | x | 1 | PF0 | |

## 2.4 Functional Description

### 2.4.1 General

Each pin except PE0, PE1, and BKGD can act as general purpose I/O. In addition each pin can act as an output from the external bus interface module or a peripheral module or an input to the external bus interface module or a peripheral module.

### 2.4.2 Registers

A set of configuration registers is common to all ports with exceptions in the expanded bus interface and ATD ports (Table 2-103). All registers can be written at any time, however a specific configuration might not become active.

**Example 2-1. Selecting a pull-up device**

```
This device does not become active while the port is used as a push-pull output.
```

**Table 2-103. Register availability per port[1]**

| Port | Data | Input | Data Direction | Reduced Drive | Pull Enable | Polarity Select | Wired-Or Mode | Interrupt Enable | Interrupt Flag | Routing |
|---|---|---|---|---|---|---|---|---|---|---|
| A | yes | - | yes | yes | yes | - | - | - | - | - |
| B | yes | - | yes | | | - | - | - | - | - |
| C | yes | - | yes | | | - | - | - | - | - |
| D | yes | - | yes | | | - | - | - | - | - |
| E | yes | - | yes | | | - | - | - | - | - |
| K | yes | - | yes | | | - | - | - | - | - |
| T | yes | yes | yes | yes | yes | yes | - | - | - | - |
| S | yes | yes | yes | yes | yes | yes | yes | - | - | yes |
| M | yes | yes | yes | yes | yes | yes | yes | - | - | yes |
| P | yes | yes | yes | yes | yes | yes | - | yes | yes | - |
| H | yes | yes | yes | yes | yes | yes | - | yes | yes | - |
| J | yes | yes | yes | yes | yes | yes | - | yes | yes | - |
| AD0 | yes | - | yes | yes | yes | - | - | - | - | - |
| AD1 | yes | - | yes | yes | yes | - | - | - | - | - |
| R | yes | yes | yes | yes | yes | yes | - | - | - | - |
| L | yes | yes | yes | yes | yes | yes | yes | - | - | yes |
| F | yes | yes | yes | yes | yes | yes | - | - | - | yes |

1. Each cell represents one register with individual configuration bits

### 2.4.2.1 Data register (PORTx, PTx)

This register holds the value driven out to the pin if the pin is used as a general purpose I/O.

Writing to this register has only an effect on the pin if the pin is used as general purpose output. When reading this address, the buffered state of the pin is returned if the associated data direction register bit is set to “0”.

If the data direction register bits are set to logic level “1”, the contents of the data register is returned. This is independent of any other configuration (Figure 2-107).

### 2.4.2.2 Input register (PTIx)

This is a read-only register and always returns the buffered state of the pin (Figure 2-107).

### 2.4.2.3 Data direction register (DDRx)

This register defines whether the pin is used as an input or an output.

If a peripheral module controls the pin the contents of the data direction register is ignored (Figure 2-107).

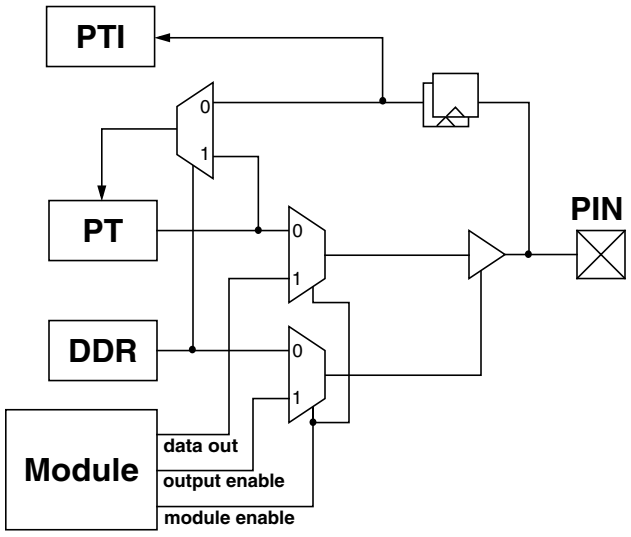



**Figure 2-107. Illustration of I/O pin functionality**

### 2.4.2.4 Reduced drive register (RDRx)

If the pin is used as an output this register allows the configuration of the drive strength.

### 2.4.2.5 Pull device enable register (PERx)

This register turns on a pull-up or pull-down device.

It becomes active only if the pin is used as an input or as a wired-or output.

### 2.4.2.6 Polarity select register (PPSx)

This register selects either a pull-up or pull-down device if enabled.

It becomes only active if the pin is used as an input. A pull-up device can be activated if the pin is used as a wired-or output.

### 2.4.2.7 Wired-or mode register (WOMx)

If the pin is used as an output this register turns off the active high drive. This allows wired-or type connections of outputs.

### 2.4.2.8 Interrupt enable register (PIEx)

If the pin is used as an interrupt input this register serves as a mask to the interrupt flag to enable/disable the interrupt.

#### 2.4.2.9 Interrupt flag register (PIFx)

If the pin is used as an interrupt input this register holds the interrupt flag after a valid pin event.

#### 2.4.2.10 Module routing register (MODRR, PTRRR, PTLRR, PTFRR)

This register supports the re-routing of the CAN0, CAN4, SPI2-0, SCI7-3, IIC0, TIM and CS[3:0] pins to alternative ports. This allows a software re-configuration of the pinouts of the different package options with respect to above peripherals.

### 2.4.3 Pins and Ports

**NOTE**

Please refer to the SOC Guide to determine the pin availability in the different package options.

#### 2.4.3.1 BKGD pin

The BKGD pin is associated with the S12X_BDM and S12X_EBI modules.

During reset, the BKGD pin is used as MODC input.

#### 2.4.3.2 Port A, B

Port A pins PA[7:0] and Port B pins PB[7:0] can be used for either general-purpose I/O with the external bus interface. In this case Port A and Port B are associated with the external address bus outputs ADDR15-ADDR8 and ADDR7-ADDR0, respectively. PB0 is the ADDR0 or $\overline{\text{UDS}}$ output.

#### 2.4.3.3 Port C, D

Port C pins PC[7:0] and Port D pins PD[7:0] can be used for either general-purpose I/O with the external bus interface. In this case Port C and Port D are associated with the external data bus inputs/outputs DATA15-DATA8 and DATA7-DATA0, respectively.

These pins are configured for reduced input threshold in certain operating modes (refer to S12X_EBI section).

#### 2.4.3.4 Port E

Port E is associated with the external bus control outputs $\overline{\text{RW}}$, $\overline{\text{LSTRB}}$, $\overline{\text{LDS}}$ and $\overline{\text{RE}}$, the free-running clock outputs ECLK and ECLK2X, as well as with the $\overline{\text{TAGHI}}$, $\overline{\text{TAGLO}}$, MODA and MODB and interrupt inputs $\overline{\text{IRQ}}$ and $\overline{\text{XIRQ}}$.

Port E pins PE[7:2] can be used for either general-purpose I/O or with the alternative functions.

Port E pin PE[7] can be used for either general-purpose I/O or as the free-running clock ECLKX2 output running at the Core Clock rate. The clock output is always enabled in emulation modes.

Port E pin PE[6] can be used for either general-purpose I/O, as $\overline{\text{TAGHI}}$ input or as MODB input during reset.

Port E pin PE[5] can be used for either general-purpose I/O, as $\overline{\text{TAGLO}}$ input, $\overline{\text{RE}}$ output or as MODA input during reset.

Port E pin PE[4] can be used for either general-purpose I/O or as the free-running clock ECLK output running at the Bus Clock rate or at the programmed divided clock rate. The clock output is always enabled in emulation modes.

Port E pin PE[3] can be used for either general-purpose I/O, as $\overline{\text{LSTRB}}$ or $\overline{\text{LDS}}$ output, or as EROMCTL input during reset.

Port E pin PE[2] can be used for either general-purpose I/O, or as $\text{R}\overline{\text{W}}$ or $\overline{\text{WE}}$ output.

Port E pin PE[1] can be used for either general-purpose input or as the level- or falling edge-sensitive $\overline{\text{IRQ}}$ interrupt input. $\overline{\text{IRQ}}$ will be enabled by setting the IRQEN configuration bit (2.3.17/119) and clearing the I-bit in the CPU condition code register. It is inhibited at reset so this pin is initially configured as a simple input with a pull-up.

Port E pin PE[0] can be used for either general-purpose input or as the level-sensitive $\overline{\text{XIRQ}}$ interrupt input. $\overline{\text{XIRQ}}$ can be enabled by clearing the X-bit in the CPU condition code register. It is inhibited at reset so this pin is initially configured as a high-impedance input with a pull-up.

Port E pins PE[5] and PE[6] are configured for reduced input threshold in certain modes (refer to S12X_EBI section).

### 2.4.3.5 Port K

Port K pins PK[7:0] can be used for either general-purpose I/O, or with the external bus interface. In this case Port K pins PK[6:0] are associated with the external address bus outputs ADDR22-ADDR16 and PK7 is associated to the $\overline{\text{EWAIT}}$ input.

Port K pin PE[7] is configured for reduced input threshold in certain modes (refer to S12X_EBI section).

### 2.4.3.6 Port T

This port is associated with the ECT module.

Port T pins PT[7:0] can be used for either general-purpose I/O, or with the channels of the Enhanced Capture Timer.

### 2.4.3.7 Port S

This port is associated with SCI0, SCI1 and SPI0.

Port S pins PS[7:4] can be used either for general-purpose I/O, or with the SPI0 subsystem.

Port S pins PS[3:2] can be used either for general-purpose I/O, or with the SCI1 subsystem.

Port S pins PS[1:0] can be used either for general-purpose I/O, or with the SCI0 subsystem.

The SPI0 pins can be re-routed.

#### 2.4.3.8 Port M

This port is associated with the SCI3 CAN4-0 and SPI0.

Port M pins PM[7:6] can be used for either general purpose I/O, or with the CAN3 subsystem.

Port M pins PM[5:4] can be used for either general purpose I/O, or with the CAN2 subsystem.

Port M pins PM[3:2] can be used for either general purpose I/O, or with the CAN1 subsystem.

Port M pins PM[1:0] can be used for either general purpose I/O, or with the CAN0 subsystem.

Port M pins PM[5:2] can be used for either general purpose I/O, or with the SPI0 subsystem.

The CAN0, CAN4 and SPI0 pins can be re-routed.

#### 2.4.3.9 Port P

This port is associated with the PWM, SPI1, SPI2 and TIM.

Port P pins PP[7:0] can be used for either general purpose I/O, or with the PWM or with the channels of the standard Timer.subsystem.

Port P pins PP[7:4] can be used for either general purpose I/O, or with the SPI2 subsystem.

Port P pins PP[3:0] can be used for either general purpose I/O, or with the SPI1 subsystem.

#### 2.4.3.10 Port H

This port is associated with the SPI1, SPI2, and SCI7-4.

Port H pins PH[7:4] can be used for either general purpose I/O, or with the SPI2 subsystem.

Port H pins PH[3:0] can be used for either general purpose I/O, or with the SPI1 subsystem.

Port H pins PH[7:6] can be used for either general purpose I/O, or with the SCI5 subsystem.

Port H pins PH[5:4] can be used for either general purpose I/O, or with the SCI4 subsystem.

Port H pins PH[3:2] can be used for either general purpose I/O, or with the SCI7 subsystem.

Port H pins PH[1:0] can be used for either general purpose I/O, or with the SCI6 subsystem.

#### 2.4.3.11 Port J

This port is associated with the chip selects $\overline{CS}$[3:0] as well as with CAN4, CAN0, IIC1, IIC0, and SCI2.

Port J pins PJ[7:6] can be used for either general purpose I/O, or with the CAN4, IIC0 or CAN0 subsystems.

Port J pins PJ[5:4] can be used for either general purpose I/O, or with the IIC1 subsystem or as chip select outputs.

Port J pin PJ[3] can be used for general purpose I/O.

Port J pin PJ[2] can be used for either general purpose I/O or as chip select output.

Port J pin PJ[1] can be used for either general purpose I/O, or with the SCI2 subsystem.

Port J pin PJ[0] can be used for either general purpose I/O, or with the SCI2 subsystem or as chip select output.

#### 2.4.3.12 Port AD0

This port is associated with the ATD0.

Port AD0 pins PAD[15:0] can be used for either general purpose I/O, or with the ATD0 subsystem.

#### 2.4.3.13 Port AD1

This port is associated with the ATD1.

Port AD1 pins PAD[31:16] can be used for either general purpose I/O, or with the ATD1 subsystem.

#### 2.4.3.14 Port R

This port is associated with the TIM module.

Port R pins PR[7:0] can be used for either general-purpose I/O, or with the channels of the standard Timer.

The TIM channels can be re-routed.

#### 2.4.3.15 Port L

This port is associated with SCI7-4.

Port L pins PL[7:6] can be used for either general purpose I/O, or with SCI7 subsystem.

Port L pins PL[5:4] can be used for either general purpose I/O, or with SCI6 subsystem.

Port L pins PL[3:2] can be used for either general purpose I/O, or with SCI5 subsystem.

Port L pins PL[1:0] can be used for either general purpose I/O, or with SCI4 subsystem.

#### 2.4.3.16 Port F

This port is associated with SCI3, IIC0 and chip selects.

Port L pins PL[7:6] can be used for either general purpose I/O, or with SCI3 subsystem.

Port L pins PL[5:4] can be used for either general purpose I/O, or with IIC0 subsystem.

Port L pins PL[3:0] can be used for either general purpose I/O, or with chip selects.

### 2.4.4 Pin interrupts

Ports P, H and J offer pin interrupt capability. The interrupt enable as well as the sensitivity to rising or falling edges can be individually configured on per-pin basis. All bits/pins in a port share the same interrupt vector. Interrupts can be used with the pins configured as inputs or outputs.

An interrupt is generated when a bit in the port interrupt flag register and its corresponding port interrupt enable bit are both set. The pin interrupt feature is also capable to wake up the CPU when it is in STOP or WAIT mode.

A digital filter on each pin prevents pulses (Figure 2-109) shorter than a specified time from generating an interrupt. The minimum time varies over process conditions, temperature and voltage (Figure 2-108 and Table 2-104).

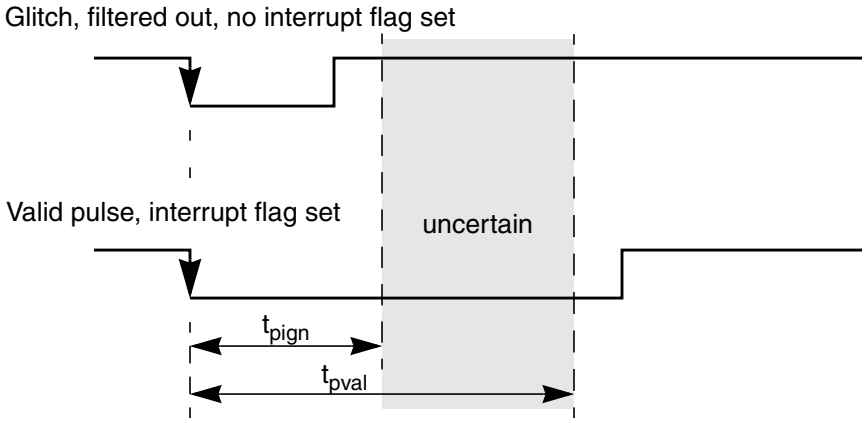



**Figure 2-108. Interrupt Glitch Filter on Port P, H and J (PPS=0)**

**Table 2-104. Pulse Detection Criteria**

| Pulse | Mode | | |
|---|---|---|---|
| | $\overline{STOP}$ | | STOP(1) |
| | | Unit | |
| Ignored | $t_{pulse} \leq 3$ | bus clocks | $t_{pulse} \leq t_{pign}$ |
| Uncertain | $3 < t_{pulse} < 4$ | bus clocks | $t_{pign} < t_{pulse} < t_{pval}$ |
| Valid | $t_{pulse} \geq 4$ | bus clocks | $t_{pulse} \geq t_{pval}$ |

1. These values include the spread of the oscillator frequency over temperature, voltage and process.

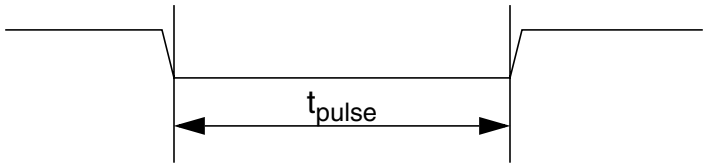



**Figure 2-109. Pulse Illustration**

A valid edge on an input is detected if 4 consecutive samples of a passive level are followed by 4 consecutive samples of an active level directly or indirectly.

The filters are continuously clocked by the bus clock in RUN and WAIT mode. In STOP mode the clock is generated by an RC-oscillator in the Port Integration Module. To maximize current saving the RC oscillator runs only if the following condition is true on any pin individually:

Sample count <= 4 and interrupt enabled (PIE=1) and interrupt flag not set (PIF=0)

## 2.5 Initialization Information

### 2.5.1 Port Data and Data Direction Register writes

It is not recommended to write PORTx/PTx and DDRx in a word access. When changing the register pins from inputs to outputs, the data may have extra transitions during the write access. Initialize the port data register before enabling the outputs.

# Chapter 3
# Memory Mapping Control (S12XMMCV4)

**Table 3-1. Revision History**

| Revision Number | Revision Date | Sections Affected | Description of Changes |
|---|---|---|---|
| V04.04 | 26 Oct 2005 | | - Reorganization of MEMCTL0 register bits. |
| V04.05 | 26 Jul 2006 | 3.4.2.4/3-212 | - Updated XGATE Memory Map |
| V04.06 | 15 Nov 2006 | | - Adding AUTOSAR Compliance concerning illegal CPU accesses |

## 3.1 Introduction

This section describes the functionality of the module mapping control (MMC) sub-block of the S12X platform. The block diagram of the MMC is shown in Figure 3-1.

The MMC module controls the multi-master priority accesses, the selection of internal resources and external space. Internal buses, including internal memories and peripherals, are controlled in this module. The local address space for each master is translated to a global memory space.

### 3.1.1 Terminology

**Table 3-2. Acronyms and Abbreviations**

| | |
|---|---|
| Logic level “1” | Voltage that corresponds to Boolean true state |
| Logic level “0” | Voltage that corresponds to Boolean false state |
| 0x | Represents hexadecimal number |
| x | Represents logic level ’don’t care’ |
| byte | 8-bit data |
| word | 16-bit data |
| local address | based on the 64 KBytes Memory Space (16-bit address) |
| global address | based on the 8 MBytes Memory Space (23-bit address) |
| Aligned address | Address on even boundary |
| Mis-aligned address | Address on odd boundary |
| Bus Clock | System Clock. Refer to CRG Block Guide. |
| expanded modes | Normal Expanded Mode<br>Emulation Single-Chip Mode<br>Emulation Expanded Mode<br>Special Test Mode |
| single-chip modes | Normal Single-Chip Mode<br>Special Single-Chip Mode |
| emulation modes | Emulation Single-Chip Mode<br>Emulation Expanded Mode |
| normal modes | Normal Single-Chip Mode<br>Normal Expanded Mode |
| special modes | Special Single-Chip Mode<br>Special Test Mode |
| NS | Normal Single-Chip Mode |
| SS | Special Single-Chip Mode |
| NX | Normal Expanded Mode |
| ES | Emulation Single-Chip Mode |
| EX | Emulation Expanded Mode |
| ST | Special Test Mode |
| Unimplemented areas | Areas which are accessible by the pages (RPAGE,PPAGE,EPAGE) and not implemented |
| External Space | Area which is accessible in the global address range 14_0000 to 3F_FFFF |
| external resource | Resources (Emulator, Application) connected to the MCU via the external bus on expanded modes (Unimplemented areas and External Space) |
| PRR | Port Replacement Registers |
| PRU | Port Replacement Unit located on the emulator side |
| MCU | MicroController Unit |
| NVM | Non-volatile Memory; Flash EEPROM or ROM |

### 3.1.2 Features

The main features of this block are:

- Paging capability to support a global 8 Mbytes memory address space
- Bus arbitration between the masters CPU, BDM and XGATE

- Simultaneous accesses to different resources[1] (internal, external, and peripherals) (see Figure 3-1 )
- Resolution of target bus access collision
- MCU operation mode control
- MCU security control
- Separate memory map schemes for each master CPU, BDM and XGATE
- ROM control bits to enable the on-chip FLASH or ROM selection
- Port replacement registers access control
- Generation of system reset when CPU accesses an unimplemented address (i.e., an address which does not belong to any of the on-chip modules) in single-chip modes

### 3.1.3 S12X Memory Mapping

The S12X architecture implements a number of memory mapping schemes including

- a CPU 8 MByte global map, defined using a global page (GPAGE) register and dedicated 23-bit address load/store instructions.
- a BDM 8 MByte global map, defined using a global page (BDMGPR) register and dedicated 23-bit address load/store instructions.
- a (CPU or BDM) 64 KByte local map, defined using specific resource page (RPAGE, EPAGE and PPAGE) registers and the default instruction set. The 64 KBytes visible at any instant can be considered as the local map accessed by the 16-bit (CPU or BDM) address.
- The XGATE 64 Kbyte local map.

The MMC module performs translation of the different memory mapping schemes to the specific global (physical) memory implementation.

### 3.1.4 Modes of Operation

This subsection lists and briefly describes all operating modes supported by the MMC.

#### 3.1.4.1 Power Saving Modes

- Run mode

  MMC is functional during normal run mode.
- Wait mode

  MMC is functional during wait mode.
- Stop mode

  MMC is inactive during stop mode.

#### 3.1.4.2 Functional Modes

- Single chip modes

  In normal and special single chip mode the internal memory is used. External bus is not active.

1. Resources are also called targets.

- Expanded modes

  Address, data, and control signals are activated in normal expanded and special test modes when accessing the external bus. Access to internal resources will not cause activity on the external bus.
- Emulation modes

  External bus is active to emulate, via an external tool, the normal expanded or the normal single chip mode.}

### 3.1.5 Block Diagram

Figure 3-1[1] shows a block diagram of the MMC.

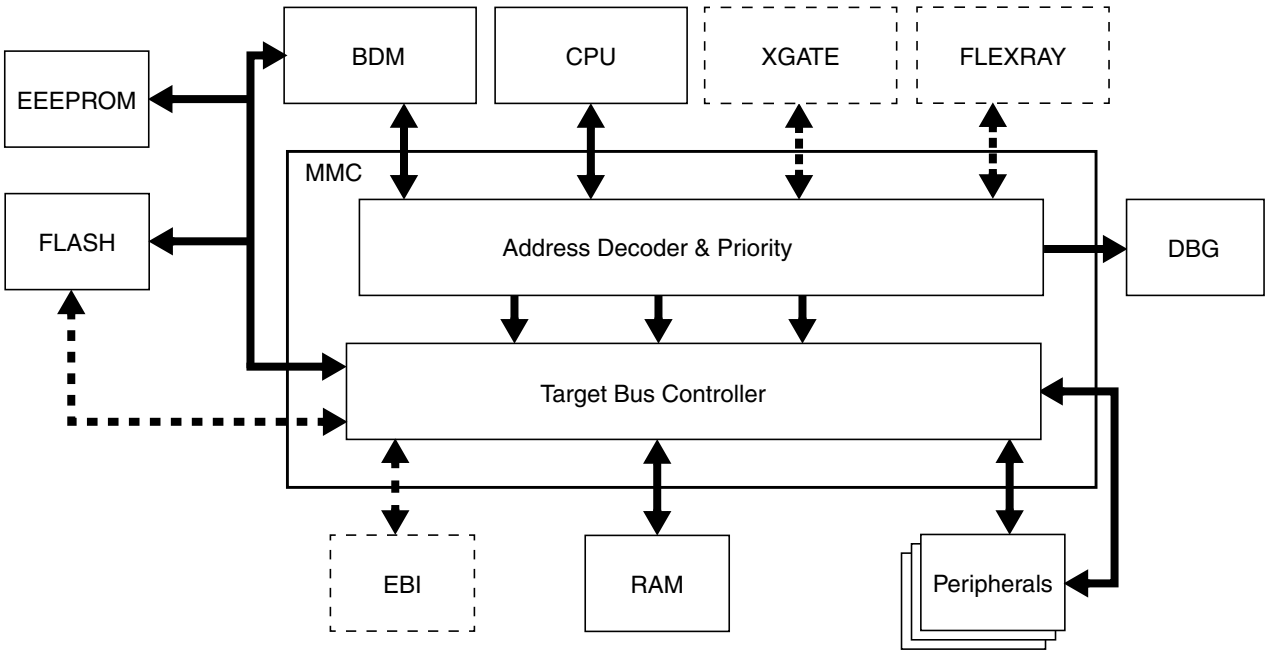



**Figure 3-1. MMC Block Diagram**

## 3.2 External Signal Description

The user is advised to refer to the device overview for port configuration and location of external bus signals. Some pins may not be bonded out in all implementations.

Table 3-3 and Table 3-4 outline the pin names and functions. It also provides a brief description of their operation.

1. Doted blocks and lines are optional. Please refer to the Device User Guide for their availlibilities.

**Table 3-3. External Input Signals Associated with the MMC**

| Signal | I/O | Description | Availability |
|---|---|---|---|
| MODC | I | Mode input | Latched after $\overline{\text{RESET}}$ (active low) |
| MODB | I | Mode input | Latched after $\overline{\text{RESET}}$ (active low) |
| MODA | I | Mode input | Latched after $\overline{\text{RESET}}$ (active low) |
| EROMCTL | I | EROM control input | Latched after $\overline{\text{RESET}}$ (active low) |
| ROMCTL | I | ROM control input | Latched after $\overline{\text{RESET}}$ (active low) |

**Table 3-4. External Output Signals Associated with the MMC**

| Signal | I/O | Description | Available in Modes | | | | | |
|---|---|---|---|---|---|---|---|---|
| | | | NS | SS | NX | ES | EX | ST |
| CS0 | O | Chip select line 0 | (see Table 3-5) | | | | | |
| CS1 | O | Chip select line 1 | | | | | | |
| CS2 | O | Chip select line 2 | | | | | | |
| CS3 | O | Chip select line 3 | | | | | | |

# 3.3 Memory Map and Registers

## 3.3.1 Module Memory Map

A summary of the registers associated with the MMC block is shown in Figure 3-2. Detailed descriptions of the registers and bits are given in the subsections that follow.

| Address | Register Name | | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|---|---|
| 0x000A | MMCCTL0 | R | CS3E1 | CS3E0 | CS2E1 | CS2E0 | CS1E1 | CS1E0 | CS0E1 | CS0E0 |
| | | W | | | | | | | | |
| 0x000B | MODE | R | MODC | MODB | MODA | 0 | 0 | 0 | 0 | 0 |
| | | W | | | | | | | | |
| 0x0010 | GPAGE | R | 0 | GP6 | GP5 | GP4 | GP3 | GP2 | GP1 | GP0 |
| | | W | | | | | | | | |
| 0x0011 | DIRECT | R | DP15 | DP14 | DP13 | DP12 | DP11 | DP10 | DP9 | DP8 |
| | | W | | | | | | | | |
| 0x0012 | Reserved | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | W | | | | | | | | |
| 0x0013 | MMCCTL1 | R | TGMRAMON | 0 | EEEIFRON | PGMIFRON | RAMHM | EROMON | ROMHM | ROMON |
| | | W | | | | | | | | |
| 0x0014 | Reserved | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | W | | | | | | | | |
| 0x0015 | PPAGE | R | PIX7 | PIX6 | PIX5 | PIX4 | PIX3 | PIX2 | PIX1 | PIX0 |
| | | W | | | | | | | | |
| 0x0016 | RPAGE | R | RP7 | RP6 | RP5 | RP4 | RP3 | RP2 | RP1 | RP0 |
| | | W | | | | | | | | |
| 0x0017 | EPAGE | R | EP7 | EP6 | EP5 | EP4 | EP3 | EP2 | EP1 | EP0 |
| | | W | | | | | | | | |

= Unimplemented or Reserved

**Figure 3-2. MMC Register Summary**

## 3.3.2 Register Descriptions

### 3.3.2.1 MMC Control Register (MMCCTL0)

Address: 0x000A PRR

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | CS3E1 | CS3E0 | CS2E1 | CS2E0 | CS1E1 | CS1E0 | CS0E1 | CS0E0 |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | $\overline{\text{ROMON}}$[1] |

1. ROMON is bit[0] of the register MMCTL1 (see Figure 3-10)

= Unimplemented or Reserved

**Figure 3-3. MMC Control Register (MMCCTL0)**

Read: Anytime. In emulation modes read operations will return the data from the external bus. In all other modes the data is read from this register.

Write: Anytime. In emulation modes write operations will also be directed to the external bus.

**Table 3-5. Chip Selects Function Activity**

| Register Bit | Chip Modes | | | | | |
|---|---|---|---|---|---|---|
| | NS | SS | NX | ES | EX | ST |
| CS0E[1:0], CS1E[1:0], CS2E[1:0], CS3E[1:0] | Disabled(1) | Disabled | Enabled(2) | Disabled | Enabled | Disabled |

1. Disabled: feature always inactive.
2. Enabled: activity is controlled by the appropriate register bit value.

The MMCCTL0 register is used to control external bus functions, like:

- Availability of chip selects. (See Table 3-5 and Table 3-6)
- Control of different external stretch mechanism. For more detail refer to the S12X_EBI BlockGuide.

**CAUTION**

XGATE write access to this register during an CPU access which makes use of this register could lead to unexpected results.

**Table 3-6. MMCCTL0 Field Descriptions**

| Field | Description |
|---|---|
| 7–6<br>CS3E[1:0] | **Chip Select 3 Enables** — These bits enable the external chip select $\overline{CS3}$ output which is asserted during accesses to specific external addresses. The associated global address range is shown in Table 3-7 and Figure 3-17.<br>Chip select 3 is only active if enabled in Normal Expanded mode, Emulation Expanded mode.<br>The function disabled in all other operating modes.<br>00 Chip select 3 is disabled<br>01,10,11 Chip select 3 is enabled |
| 5–4<br>CS2E[1:0] | **Chip Select 2 Enables** — These bits enable the external chip select $\overline{CS2}$ output which is asserted during accesses to specific external addresses. The associated global address range is shown in Table 3-7 and Figure 3-17.<br>Chip select 2 is only active if enabled in Normal Expanded mode, Emulation Expanded mode.<br>The function disabled in all other operating modes.<br>00 Chip select 2 is disabled<br>01,10,11 Chip select 2 is enabled |
| 3–2<br>CS1E[1:0] | **Chip Select 1 Enables** — These bits enable the external chip select $\overline{CS1}$ output which is asserted during accesses to specific external addresses. The associated global address range is shown in Table 3-7 and Figure 3-17.<br>Chip select 1 is only active if enabled in Normal Expanded mode, Emulation Expanded mode.<br>The function disabled in all other operating modes.<br>00 Chip select 1 is disabled<br>01,10,11 Chip select 1 is enabled |
| 1–0<br>CS0E[1:0] | **Chip Select 0 Enables** — These bits enable the external chip select $\overline{CS0}$ output which is asserted during accesses to specific external addresses. The associated global address range is shown in Table 3-7 and Figure 3-17.<br>Chip select 0 is only active if enabled in Normal Expanded mode, Emulation Expanded mode.<br>The function disabled in all other operating modes.<br>00 Chip select 0 is disabled<br>01,10,11 Chip select 0 is enabled |

Table 3-7 shows the address boundaries of each chip select and the relationship with the implemented resources (internal) parameters.

**Table 3-7. Global Chip Selects Memory Space**

| Chip Selects | Bottom Address | Top Address |
|---|---|---|
| $\overline{CS3}$ | 0x00_0800 | 0x0F_FFFF minus RAMSIZE(1) |
| $\overline{CS2}$(2) | 0x14_0000 | 0x1F_FFFF |
| $\overline{CS1}$ | 0x20_0000 | 0x3F_FFFF |
| $\overline{CS0}$(3) | 0x40_0000 | 0x7F_FFFF minus FLASHSIZE(4) |

1. External RPAGE accesses in (NX, EX)
2. When ROMHM is set (see ROMHM in Table 3-16) the $\overline{CS2}$ is asserted in the space occupied by this on-chip memory block.
3. When the internal NVM is enabled (see ROMON in Section 3.3.2.5, “MMC Control Register (MMCCTL1)) the $\overline{CS0}$ is not asserted in the space occupied by this on-chip memory block.
4. External PPAGE accesses in (NX, EX)

### 3.3.2.2 Mode Register (MODE)

Address: 0x000B PRR

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | MODC | MODB | MODA | 0 | 0 | 0 | 0 | 0 |
| W | | | | | | | | |
| Reset | MODC[1] | MODB[1] | MODA[1] | 0 | 0 | 0 | 0 | 0 |

1. External signal (see Table 3-3).

= Unimplemented or Reserved

**Figure 3-4. Mode Register (MODE)**

Read: Anytime. In emulation modes read operations will return the data read from the external bus. In all other modes the data are read from this register.

Write: Only if a transition is allowed (see Figure 3-5). In emulation modes write operations will be also directed to the external bus.

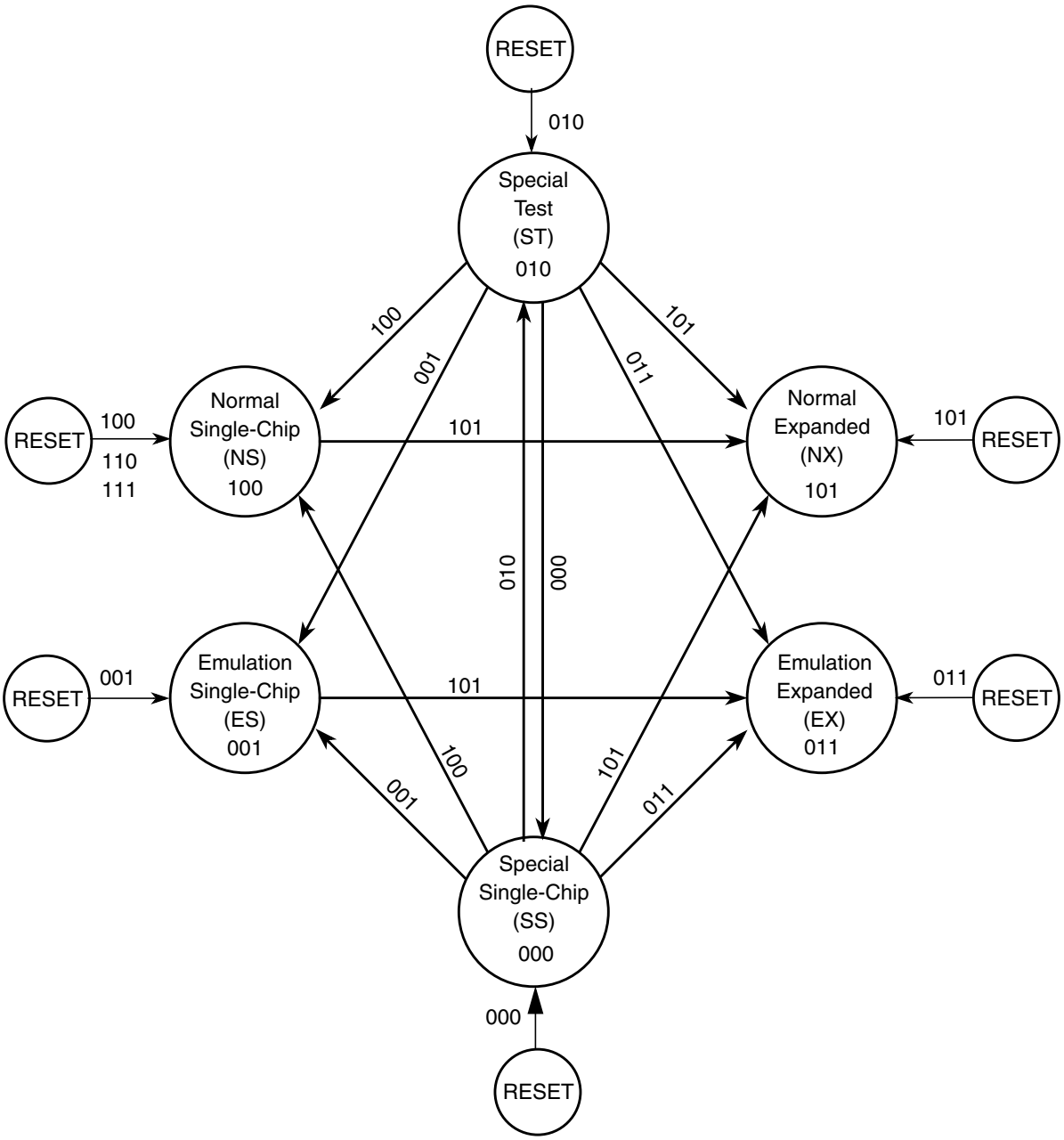
The MODE bits of the MODE register are used to establish the MCU operating mode.

**CAUTION**

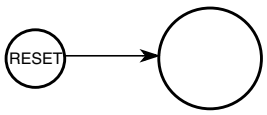
XGATE write access to this register during an CPU access which makes use of this register could lead to unexpected results.

**Table 3-8. MODE Field Descriptions**

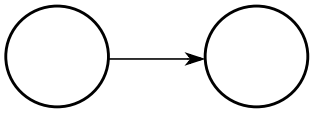
| Field | Description |
|---|---|
| 7–5<br>MODC,<br>MODB,<br>MODA | **Mode Select Bits** — These bits control the current operating mode during $\overline{\text{RESET}}$ high (inactive). The external mode pins MODC, MODB, and MODA determine the operating mode during $\overline{\text{RESET}}$ low (active). The state of the pins is latched into the respective register bits after the $\overline{\text{RESET}}$ signal goes inactive (see Figure 3-4).<br>Write restrictions exist to disallow transitions between certain modes. Figure 3-5 illustrates all allowed mode changes. Attempting non authorized transitions will not change the MODE bits, but it will block further writes to these register bits except in special modes.<br>Both transitions from normal single-chip mode to normal expanded mode and from emulation single-chip to emulation expanded mode are only executed by writing a value of 3'b101 (write once). Writing any other value will not change the MODE bits, but will block further writes to these register bits.<br>Changes of operating modes are not allowed when the device is secured, but it will block further writes to these register bits except in special modes.<br>In emulation modes reading this address returns data from the external bus which has to be driven by the emulator. It is therefore responsibility of the emulator hardware to provide the expected value (i.e. a value corresponding to normal single chip mode while the device is in emulation single-chip mode or a value corresponding to normal expanded mode while the device is in emulation expanded mode). |




Transition done by external pins (MODC, MODB, MODA)




Transition done by write access to the MODE register



110
111

Illegal (MODC, MODB, MODA) pin values.

Do not use. (Reserved for future use).

**Figure 3-5. Mode Transition Diagram when MCU is Unsecured**

### 3.3.2.3 Global Page Index Register (GPAGE)

Address: 0x0010

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | 0 | GP6 | GP5 | GP4 | GP3 | GP2 | GP1 | GP0 |
| W | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented or Reserved

**Figure 3-6. Global Page Index Register (GPAGE)**

Read: Anytime

Write: Anytime

The global page index register is used to construct a 23 bit address in the global map format. It is only used when the CPU is executing a global instruction (GLDAA, GLDAB, GLDD, GLDS, GLDX, GLDY,GSTAA, GSTAB, GSTD, GSTS, GSTX, GSTY) (see CPU Block Guide). The generated global address is the result of concatenation of the CPU local address [15:0] with the GPAGE register [22:16] (see Figure 3-7).

**CAUTION**

XGATE write access to this register during an CPU access which makes use of this register could lead to unexpected results.

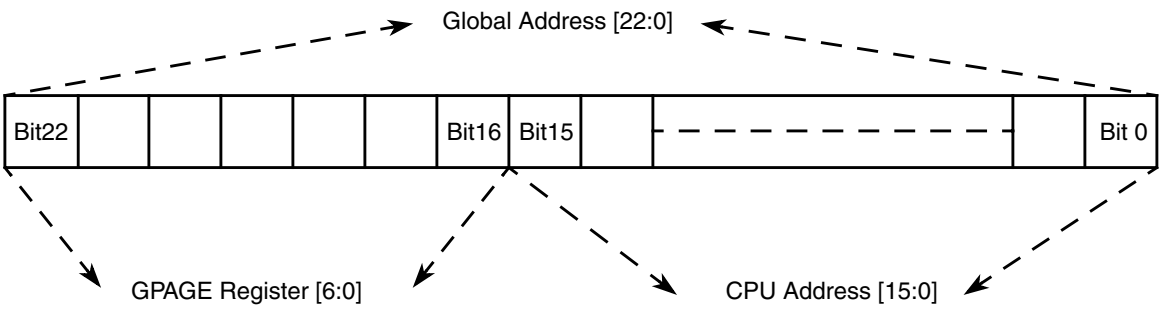



**Figure 3-7. GPAGE Address Mapping**

**Table 3-9. GPAGE Field Descriptions**

| Field | Description |
|---|---|
| 6–0<br>GP[6:0] | **Global Page Index Bits 6–0** — These page index bits are used to select which of the 128 64-kilobyte pages is to be accessed. |

**Example 3-1. This example demonstrates usage of the GPAGE register**

```
LDX     #0x5000          ;Set GPAGE offset to the value of 0x5000
MOVB    #0x14, GPAGE     ;Initialize GPAGE register with the value of 0x14
GLDAA   X                ;Load Accu A from the global address 0x14_5000
```

### 3.3.2.4 Direct Page Register (DIRECT)

Address: 0x0011

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | DP15 | DP14 | DP13 | DP12 | DP11 | DP10 | DP9 | DP8 |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 3-8. Direct Register (DIRECT)**

Read: Anytime

Write: anytime in special modes, one time only in other modes.

This register determines the position of the 256 Byte direct page within the memory map.It is valid for both global and local mapping scheme.

**Table 3-10. DIRECT Field Descriptions**

| Field | Description |
|---|---|
| 7–0<br>DP[15:8] | **Direct Page Index Bits 15–8** — These bits are used by the CPU when performing accesses using the direct addressing mode. The bits from this register form bits [15:8] of the address (see Figure 3-9). |

**CAUTION**

XGATE write access to this register during an CPU access which makes use of this register could lead to unexpected results.

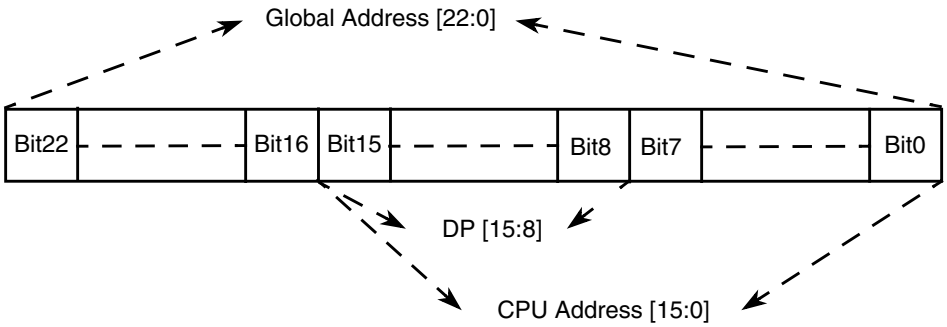



**Figure 3-9. DIRECT Address Mapping**

Bits [22:16] of the global address will be formed by the GPAGE[6:0] bits in case the CPU executes a global instruction in direct addressing mode or by the appropriate local address to the global address expansion (refer to Section 3.4.2.1.1, “Expansion of the Local Address Map).

**Example 3-2. This example demonstrates usage of the Direct Addressing Mode**

```
MOVB     #0x80,DIRECT      ;Set DIRECT register to 0x80. Write once only.
                           ;Global data accesses to the range 0xXX_80XX can be direct.
                           ;Logical data accesses to the range 0x80XX are direct.

LDY      <00               ;Load the Y index register from 0x8000 (direct access).
                           ;< operator forces direct access on some assemblers but in
```

```
                                ;many cases assemblers are "direct page aware" and can
                                ;automatically select direct mode.
```

### 3.3.2.5 MMC Control Register (MMCCTL1)

Address: 0x0013 PRR

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | TGMRAMON | 0 | EEEIFRON | PGMIFRON | RAMHM | EROMON | ROMHM | ROMON |
| W | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | EROMCTL | 0 | ROMCTL |

= Unimplemented or Reserved

**Figure 3-10. MMC Control Register (MMCCTL1)**

Read: Anytime. In emulation modes read operations will return the data from the external bus. In all other modes the data are read from this register.

Write: Refer to each bit description. In emulation modes write operations will also be directed to the external bus.

**CAUTION**

XGATE write access to this register during an CPU access which makes use of this register could lead to unexpected results.

**Table 3-11. MMCCTL1 Field Descriptions**

| Field | Description |
|---|---|
| 7<br>TGMRAMON | **EEE Tag RAM and FTM SCRATCH RAM visible in the memory map**<br>Write: Anytime<br>This bit is used to made the EEE Tag RAM nd FTM SCRATCH RAM visible in the global memory map.<br>0 Not visible in the memory map.<br>1 Visible in the memory map. |
| 5<br>EEEIFRON | **EEE IFR visible in the memory map**<br>Write: Anytime<br>This bit is used to made the IFR sector of EEE DATA FLASH visible in the global memory map.<br>0 Not visible in the memory map.<br>1 Visible in the memory map. |
| 4<br>PGMIFRON | **Program IFR visible in the memory map**<br>Write: Anytime<br>This bit is used to made the IFR sector of the Program Flash visible in the global memory map.<br>0 Not visible in the memory map.<br>1 Visible in the memory map. |
| 3<br>RAMHM | **RAM only in higher Half of the memory map**<br>Write: Once in normal and emulation modes and anytime in special modes<br>0 Accesses to $4000–$7FFF will be mapped to $14_4000-$14_7FFF in the global memory space (external access).<br>1 Accesses to $4000–$7FFF will be mapped to $0F_C000-$0F_FFFF in the global memory space (RAM area). |

**Table 3-11. MMCCTL1 Field Descriptions (continued)**

| Field | Description |
|---|---|
| 2<br>EROMON | **Enables emulated Flash or ROM memory in the memory map**<br>Write: Never<br>This bit is used in some modes to define the placement of the Emulated Flash or ROM (Refer to Table 3-12)<br>0 Disables the emulated Flash or ROM in the memory map.<br>1 Enables the emulated Flash or ROM in the memory map. |
| 1<br>ROMHM | **FLASH or ROM only in higher Half of Memory Map**<br>Write: Once in normal and emulation modes and anytime in special modes<br>0 The fixed page of Flash or ROM can be accessed in the lower half of the memory map. Accesses to 0x4000–0x7FFF will be mapped to 0x7F_4000-0x7F_7FFF in the global memory space.<br>1 Disables access to the Flash or ROM in the lower half of the memory map.These physical locations of the Flash or ROM can still be accessed through the program page window. Accesses to 0x4000–0x7FFF will be mapped to 0x14_4000-0x14_7FFF in the global memory space (external access). |
| 0<br>ROMON | **Enable FLASH or ROM in the memory map**<br>Write: Once in normal and emulation modes and anytime in special modes.<br>This bit is used in some modes to define the placement of the ROM (Refer to Table 3-12)<br>0 Disables the Flash or ROM from the memory map.<br>1 Enables the Flash or ROM in the memory map. |

EROMON and ROMON control the visibility of the Flash in the memory map for CPU or BDM (not for XGATE). Both local and global memory maps are affected.

**Table 3-12. Data Sources when CPU or BDM is Accessing Flash Area**

| Chip Modes | ROMON | EROMON | DATA SOURCE(1) | Stretch(2) |
|---|---|---|---|---|
| Normal Single Chip | X | X | Internal Flash | N |
| Special Single Chip | | | | |
| Emulation Single Chip | X | 0 | Emulation Memory | N |
| | X | 1 | Internal Flash | |
| Normal Expanded | 0 | X | External Application | Y |
| | 1 | X | Internal Flash | N |
| Emulation Expanded | 0 | X | External Application | Y |
| | 1 | 0 | Emulation Memory | N |
| | 1 | 1 | Internal Flash | |
| Special Test | 0 | X | External Application | N |
| | 1 | X | Internal Flash | |

1. Internal Flash means Flash resources inside the MCU are read/written.
   Emulation memory means resources inside the emulator are read/written (PRU registers, flash replacement, RAM, EEPROM and register space are always considered internal).
   External application means resources residing outside the MCU are read/written.
2. The external access stretch mechanism is part of the EBI module (refer to EBI Block Guide for details).

### 3.3.2.6 Program Page Index Register (PPAGE)

Address: 0x0015

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | PIX7 | PIX6 | PIX5 | PIX4 | PIX3 | PIX2 | PIX1 | PIX0 |
| Reset | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 |

**Figure 3-11. Program Page Index Register (PPAGE)**

Read: Anytime

Write: Anytime

These eight index bits are used to page 16 KByte blocks into the Flash page window located in the local (CPU or BDM) memory map from address 0x8000 to address 0xBFFF (see Figure 3-12). This supports accessing up to 4 Mbytes of Flash (in the Global map) within the 64 KByte Local map. The PPAGE register is effectively used to construct paged Flash addresses in the Local map format. The CPU has special access to read and write this register directly during execution of CALL and RTC instructions..

**CAUTION**

XGATE write access to this register during an CPU access which makes use of this register could lead to unexpected results.

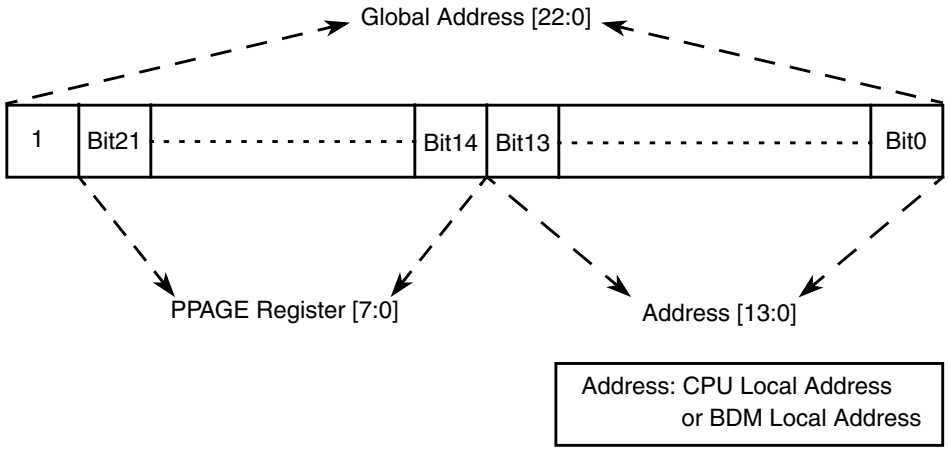



**Figure 3-12. PPAGE Address Mapping**

**NOTE**

Writes to this register using the special access of the CALL and RTC instructions will be complete before the end of the instruction execution.

**Table 3-13. PPAGE Field Descriptions**

| Field | Description |
|---|---|
| 7–0<br>PIX[7:0] | **Program Page Index Bits 7–0** — These page index bits are used to select which of the 256 FLASH or ROM array pages is to be accessed in the Program Page Window. |

The fixed 16K page from 0x4000–0x7FFF (when ROMHM = 0) is the page number 0xFD.

The reset value of 0xFE ensures that there is linear Flash space available between addresses 0x4000 and 0xFFFF out of reset.

The fixed 16K page from 0xC000-0xFFFF is the page number 0xFF.

### 3.3.2.7 RAM Page Index Register (RPAGE)

Address: 0x0016

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | RP7 | RP6 | RP5 | RP4 | RP3 | RP2 | RP1 | RP0 |
| Reset | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 1 |

**Figure 3-13. RAM Page Index Register (RPAGE)**

Read: Anytime

Write: Anytime

These eight index bits are used to page 4 KByte blocks into the RAM page window located in the local (CPU or BDM) memory map from address 0x1000 to address 0x1FFF (see Figure 3-14). This supports accessing up to 1022 KByte of RAM (in the Global map) within the 64 KByte Local map. The RAM page index register is effectively used to construct paged RAM addresses in the Local map format.

**CAUTION**

XGATE write access to this register during an CPU access which makes use of this register could lead to unexpected results.

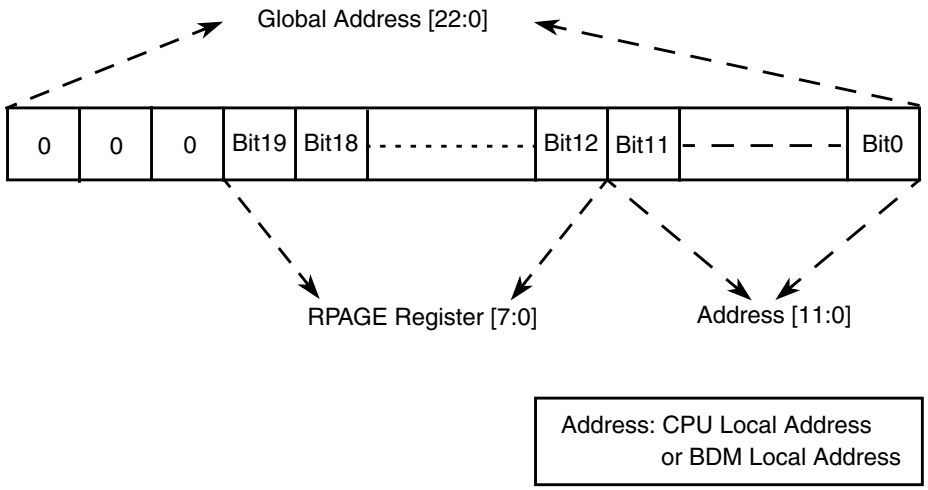



**Figure 3-14. RPAGE Address Mapping**

**NOTE**

Because RAM page 0 has the same global address as the register space, it is possible to write to registers through the RAM space when RPAGE = 0x00.

**Table 3-14. RPAGE Field Descriptions**

| Field | Description |
|---|---|
| 7–0<br>RP[7:0] | **RAM Page Index Bits 7–0** — These page index bits are used to select which of the 256 RAM array pages is to be accessed in the RAM Page Window. |

The reset value of 0xFD ensures that there is a linear RAM space available between addresses 0x1000 and 0x3FFF out of reset.

The fixed 4K page from 0x2000–0x2FFF of RAM is equivalent to page 254 (page number 0xFE).

The fixed 4K page from 0x3000–0x3FFF of RAM is equivalent to page 255 (page number 0xFF).

### 3.3.2.8 EEPROM Page Index Register (EPAGE)

Address: 0x0017

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | EP7 | EP6 | EP5 | EP4 | EP3 | EP2 | EP1 | EP0 |
| Reset | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 |

**Figure 3-15. EEPROM Page Index Register (EPAGE)**

Read: Anytime

Write: Anytime

These eight index bits are used to page 1 KByte blocks into the EEPROM page window located in the local (CPU or BDM) memory map from address 0x0800 to address 0x0BFF (see Figure 3-16). This supports accessing up to 256 KByte of EEPROM (in the Global map) within the 64 KByte Local map. The EEPROM page index register is effectively used to construct paged EEPROM addresses in the Local map format.

**CAUTION**

XGATE write access to this register during an CPU access which makes use of this register could lead to unexpected results.

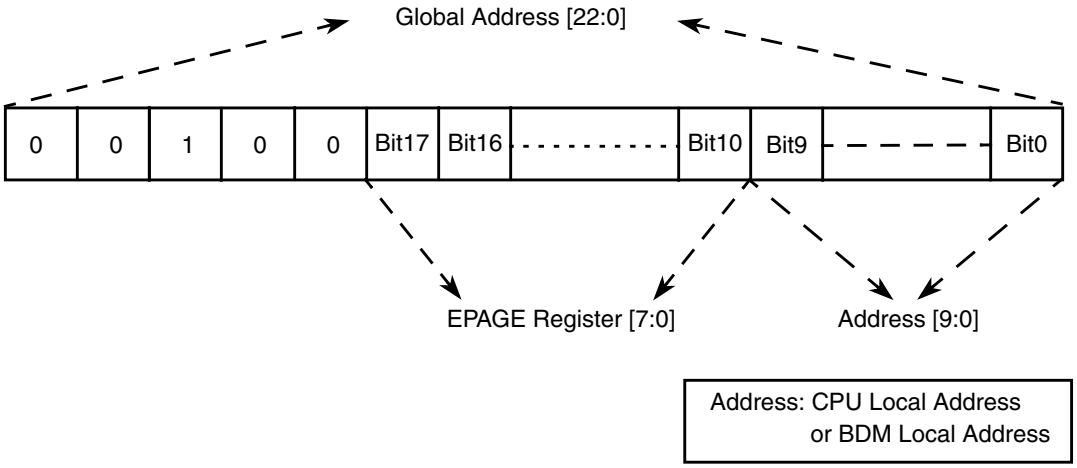



**Figure 3-16. EPAGE Address Mapping**

**Table 3-15. EPAGE Field Descriptions**

| Field | Description |
|---|---|
| 7–0<br>EP[7:0] | **EEPROM Page Index Bits 7–0** — These page index bits are used to select which of the 256 EEPROM array pages is to be accessed in the EEPROM Page Window. |

The reset value of 0xFE ensures that there is a linear EEPROM space available between addresses 0x0800 and 0x0FFF out of reset.

The fixed 1K page 0x0C00–0x0FFF of EEPROM is equivalent to page 255 (page number 0xFF).

## 3.4 Functional Description

The MMC block performs several basic functions of the S12X sub-system operation: MCU operation modes, priority control, address mapping, select signal generation and access limitations for the system. Each aspect is described in the following subsections.

### 3.4.1 MCU Operating Mode

- Normal single-chip mode

  There is no external bus in this mode. The MCU program is executed from the internal memory and no external accesses are allowed.
- Special single-chip mode

  This mode is generally used for debugging single-chip operation, boot-strapping or security related operations. The active background debug mode is in control of the CPU code execution and the BDM firmware is waiting for serial commands sent through the BKGD pin. There is no external bus in this mode.

- Emulation single-chip mode

  Tool vendors use this mode for emulation systems in which the user's target application is normal single-chip mode. Code is executed from external or internal memory depending on the set-up of the EROMON bit (see Section 3.3.2.5, "MMC Control Register (MMCCTL1)). The external bus is active in both cases to allow observation of internal operations (internal visibility).
- Normal expanded mode

  The external bus interface is configured as an up to 23-bit address bus, 8 or 16-bit data bus with dedicated bus control and status signals. This mode allows 8 or 16-bit external memory and peripheral devices to be interfaced to the system. The fastest external bus rate is half of the internal bus rate. An external signal can be used in this mode to cause the external bus to wait as desired by the external logic.
- Emulation expanded mode

  Tool vendors use this mode for emulation systems in which the user's target application is normal expanded mode.
- Special test mode

  This mode is an expanded mode for factory test.

## 3.4.2 Memory Map Scheme

### 3.4.2.1 CPU and BDM Memory Map Scheme

The BDM firmware lookup tables and BDM register memory locations share addresses with other modules; however they are not visible in the memory map during user's code execution. The BDM memory resources are enabled only during the READ_BD and WRITE_BD access cycles to distinguish between accesses to the BDM memory area and accesses to the other modules. (Refer to BDM Block Guide for further details).

When the MCU enters active BDM mode, the BDM firmware lookup tables and the BDM registers become visible in the local memory map in the range 0xFF00-0xFFFF (global address 0x7F_FF00 - 0x7F_FFFF) and the CPU begins execution of firmware commands or the BDM begins execution of hardware commands. The resources which share memory space with the BDM module will not be visible in the memory map during active BDM mode.

Please note that after the MCU enters active BDM mode the BDM firmware lookup tables and the BDM registers will also be visible between addresses 0xBF00 and 0xBFFF if the PPAGE register contains value of 0xFF.

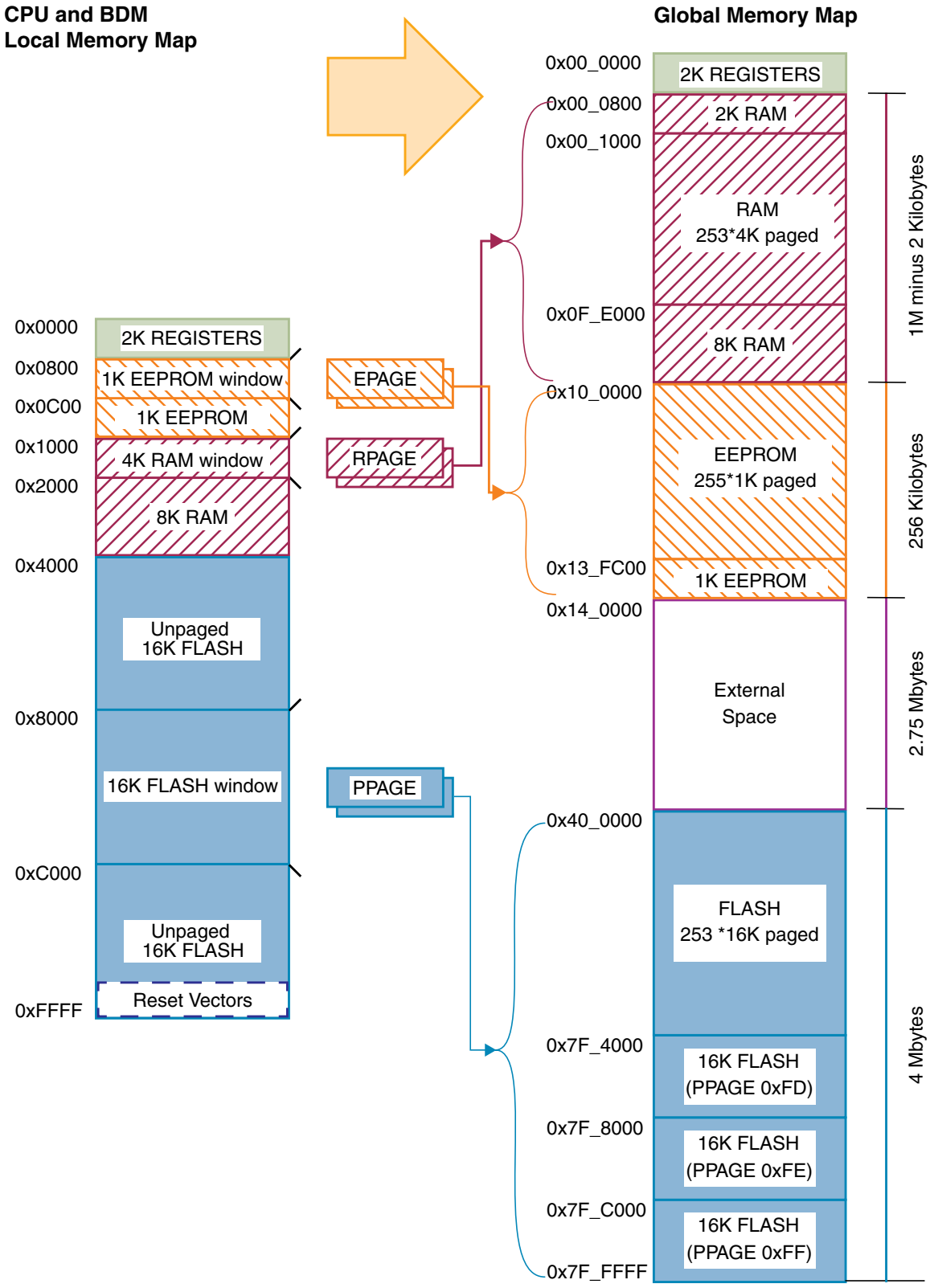



**Figure 3-17. Expansion of the Local Address Map**

#### 3.4.2.1.1 Expansion of the Local Address Map

### Expansion of the CPU Local Address Map

The program page index register in MMC allows accessing up to 4 Mbyte of FLASH or ROM in the global memory map by using the eight page index bits to page 256 16 Kbyte blocks into the program page window located from address 0x8000 to address 0xBFFF in the local CPU memory map.

The page value for the program page window is stored in the PPAGE register. The value of the PPAGE register can be read or written by normal memory accesses as well as by the CALL and RTC instructions (see Section 3.5.1, “CALL and RTC Instructions).

Control registers, vector space and parts of the on-chip memories are located in unpaged portions of the 64-kilobyte local CPU address space.

The starting address of an interrupt service routine must be located in unpaged memory unless the user is certain that the PPAGE register will be set to the appropriate value when the service routine is called. However an interrupt service routine can call other routines that are in paged memory. The upper 16-kilobyte block of the local CPU memory space (0xC000–0xFFFF) is unpaged. It is recommended that all reset and interrupt vectors point to locations in this area or to the other unpaged sections of the local CPU memory map.

Table 3-16 summarizes mapping of the address bus in Flash/External space based on the address, the PPAGE register value and value of the ROMHM bit in the MMCCTL1 register.

**Table 3-16. Global FLASH/ROM Allocated**

| Local CPU Address | ROMHM | External Access | Global Address |
|---|---|---|---|
| 0x4000–0x7FFF | 0 | No | 0x7F_4000 –0x7F_7FFF |
| | 1 | Yes | 0x14_4000–0x14_7FFF |
| 0x8000–0xBFFF | N/A | No(1) | 0x40_0000–0x7F_FFFF |
| | N/A | Yes[1] | |
| 0xC000–0xFFFF | N/A | No | 0x7F_C000–0x7F_FFFF |

1. The internal or the external bus is accessed based on the size of the memory resources implemented on-chip. Please refer to Figure 1-23 for further details.

The RAM page index register allows accessing up to 1 Mbyte –2 Kbytes of RAM in the global memory map by using the eight RPAGE index bits to page 4 Kbyte blocks into the RAM page window located in the local CPU memory space from address 0x1000 to address 0x1FFF. The EEPROM page index register EPAGE allows accessing up to 256 Kbytes of EEPROM in the system by using the eight EPAGE index bits to page 1 Kbyte blocks into the EEPROM page window located in the local CPU memory space from address 0x0800 to address 0x0BFF.

### Expansion of the BDM Local Address Map

PPAGE, RPAGE, and EPAGE registers are also used for the expansion of the BDM local address to the global address. These registers can be read and written by the BDM.

The BDM expansion scheme is the same as the CPU expansion scheme.

## 3.4.2.2 Global Addresses Based on the Global Page

### CPU Global Addresses Based on the Global Page

The seven global page index bits allow access to the full 8 Mbyte address map that can be accessed with 23 address bits. This provides an alternative way to access all of the various pages of FLASH, RAM and EEE as well as additional external memory.

The GPAGE Register is used only when the CPU is executing a global instruction (see Section 3.3.2.3, “Global Page Index Register (GPAGE)). The generated global address is the result of concatenation of the CPU local address [15:0] with the GPAGE register [22:16] (see Figure 3-7).

### BDM Global Addresses Based on the Global Page

The seven BDMGPR Global Page index bits allow access to the full 8 Mbyte address map that can be accessed with 23 address bits. This provides an alternative way to access all of the various pages of FLASH, RAM and EEE as well as additional external memory.

The BDM global page index register (BDMGPR) is used only in the case the CPU is executing a firmware command which uses a global instruction (like GLDD, GSTD) or by a BDM hardware command (like WRITE_W, WRITE_BYTE, READ_W, READ_BYTE). See the BDM Block Guide for further details.

The generated global address is a result of concatenation of the BDM local address with the BDMGPR register [22:16] in the case of a hardware command or concatenation of the CPU local address and the BDMGPR register [22:16] in the case of a firmware command (see Figure 3-18).

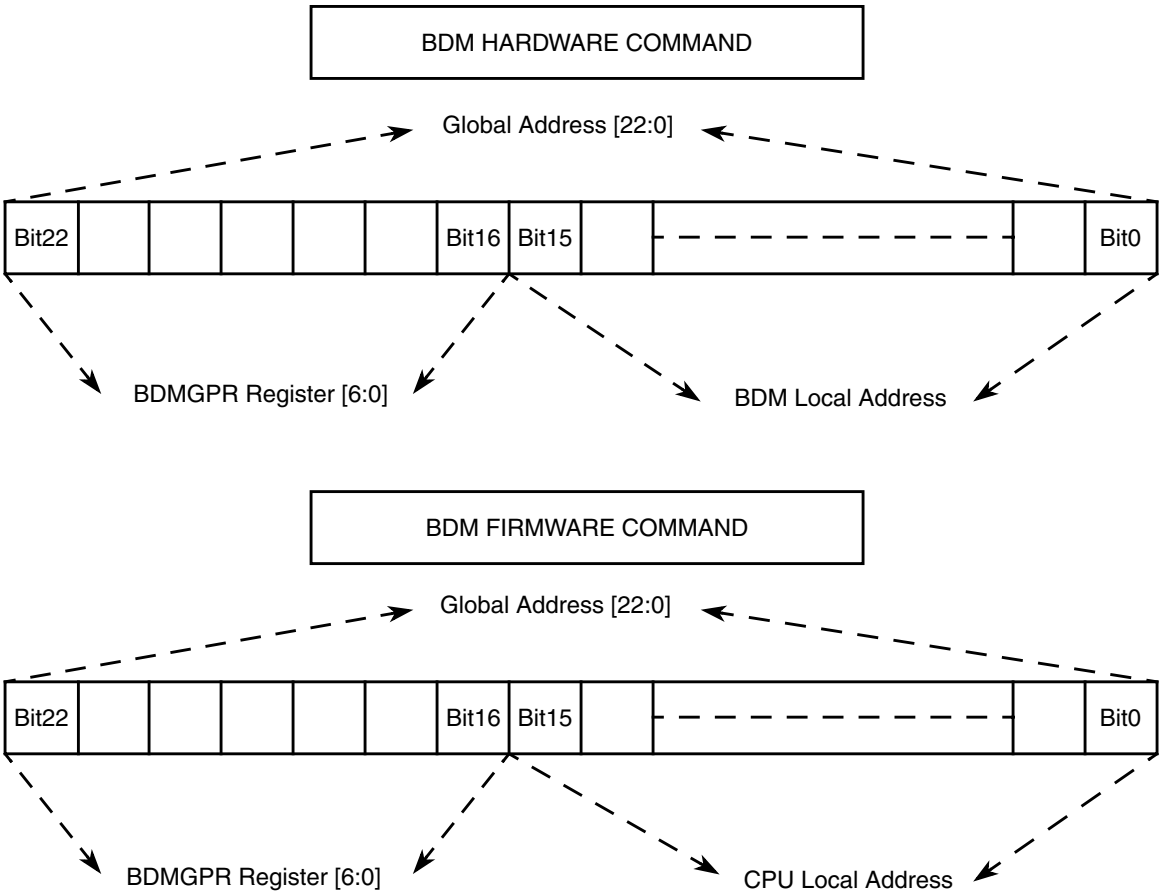



**Figure 3-18. BDMGPR Address Mapping**

### 3.4.2.3 Implemented Memory Map

The global memory spaces reserved for the internal resources (RAM, EEE, and FLASH) are not determined by the MMC module. Size of the individual internal resources are however fixed in the design of the device cannot be changed by the user. Please refer to the Device User Guide for further details. Figure 3-19 and Table 3-17 show the memory spaces occupied by the on-chip resources. Please note that the memory spaces have fixed top addresses.

**Table 3-17. Global Implemented Memory Space**

| Internal Resource | $Address |
|---|---|
| RAM | RAM_LOW = 0x10_0000 minus RAMSIZE(1) |
| FLASH | FLASH_LOW = 0x80_0000 minus FLASHSIZE(2) |

1. RAMSIZE is the hexadecimal value of RAM SIZE in bytes
2. FLASHSIZE is the hexadecimal value of FLASH SIZE in bytes

When the device is operating in expanded modes except emulation single-chip mode, accesses to global addresses which are not occupied by the on-chip resources (unimplemented areas or external memory space) result in accesses to the external bus (see Figure 3-19).

In emulation single-chip mode, accesses to global addresses which are not occupied by the on-chip resources (unimplemented areas) result in accesses to the external bus. CPU accesses to global addresses which are occupied by external memory space result in an illegal access reset (system reset) in case of no MPU error. BDM accesses to the external space are performed but the data will be undefined.

In single-chip modes accesses by the CPU (except for firmware commands) to any of the unimplemented areas (see Figure 3-19) will result in an illegal access reset (system reset) in case of no MPU error. BDM accesses to the unimplemented areas are allowed but the data will be undefined.

No misaligned word access from the BDM module will occur; these accesses are blocked in the BDM module (Refer to BDM Block Guide).

Misaligned word access to the last location of RAM is performed but the data will be undefined.

Misaligned word access to the last location of any global page (64 Kbyte) by any global instruction, is performed by accessing the last byte of the page and the first byte of the same page, considering the above mentioned misaligned access cases.

The non-internal resources (unimplemented areas or external space) are used to generate the chip selects (CS0,CS1,CS2 and CS3) (see Figure 3-19), which are only active in normal expanded, emulation expanded (see Section 3.3.2.1, “MMC Control Register (MMCCTL0)).

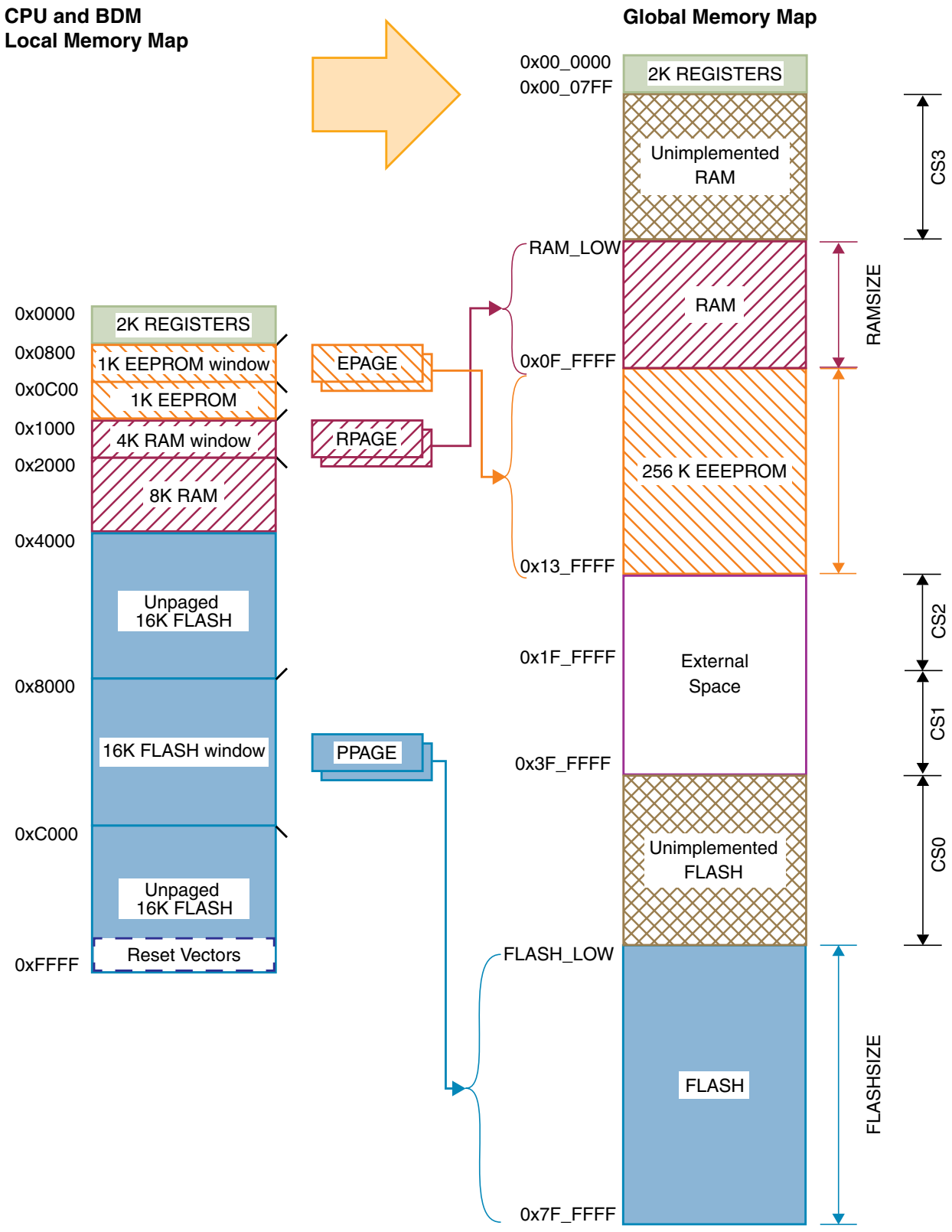



**Figure 3-19. S12XE CPU & BDM Global Address Mapping**

### 3.4.2.4 XGATE Memory Map Scheme

#### 3.4.2.4.1 Expansion of the XGATE Local Address Map

The XGATE 64 Kbyte memory space allows access to internal resources only (Registers, RAM, and FLASH). The 2 Kilobyte register address range is the same register address range as for the CPU and the BDM module (see Table 3-18).

XGATE can access the FLASH in single chip modes, even when the MCU is secured. In expanded modes, XGATE can not access the FLASH when MCU is secured.

The local address of the XGATE RAM access is translated to the global RAM address range. The XGATE shares the RAM resource with the CPU and the BDM module (see Table 3-18).

XGATE RAM size (XGRAMSIZE) may be lower or equal to the MCU RAM size (RAMSIZE).In case of XGATE RAM size less than 32 Kbytes (see Figure 3-20), the gap in the xgate local memory map will result in an illegal RAM access (see Section 3.4.3.1, “Illegal XGATE Accesses)

The local address of the XGATE FLASH access is always translated to the global address 0x78_0800 - 0x78_7FFF.

Example 3-3. is a general example of the XGATE memory map implementation.

**Table 3-18. XGATE Implemented Memory Space**

| Internal Resource | $Address |
|---|---|
| XGATE RAM | XGRAM_LOW = 0x0F_0000 plus (0x1_0000 minus XGRAMSIZE)[1] |

1. XGRAMSIZE is the hexadecimal value of XGATE RAM SIZE in bytes.

**Example 3-3.**

The MCU FLASHSIZE is 64 Kbytes (0x10000) and MCU RAMSIZE is 32 Kbytes (0x8000).

The XGATE RAMSIZE is 16 Kbytes (0x4000).

The space occupied by the XGATE RAM in the global address space will be:

Bottom address: (0x10_0000 minus 0x4000) = 0x0F_C000

Top address: 0x0F_FFFF

XGATE accesses to local address range 0x0800–0x7FFF will result always in accesses to the following FLASH block in the global address space:

Bottom address: 0x78_0800

Top address: 0x78_7FFF

The gap range in the local memory map 0x8000–0xBFFF will be translated in the global address space:

0x0F_8000 - 0x0F_BFFF (illegal xgate access to system RAM).

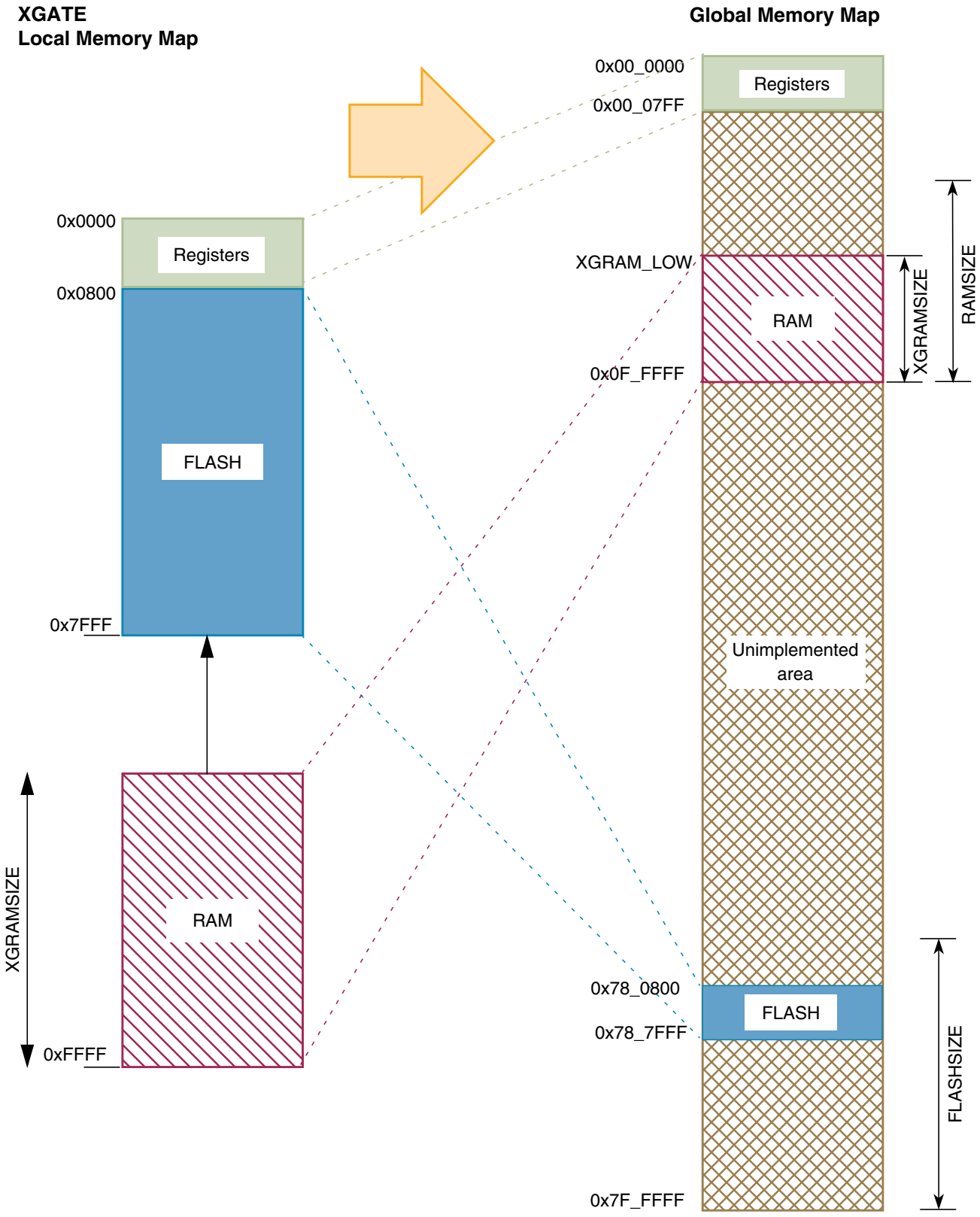



**Figure 3-20. XGATE Global Address Mapping**

### 3.4.2.5 Memory Configuration

Two bits in the MMCCTL1 register (ROMHM, RAMHM) configure the mapping of the local address (0x4000-0x7FFF) in the global memory map.

ROMHM, RAMHM are write once in normal and emulation modes and anytime in special modes.

Three areas are identified (See Figure 3-21):

- Program FLASH (0x7F_4000-0x7F_7FFF) when ROMHM = 0.
- External Space (0x14_4000-0x14_7FFF) when ROMHM = 1 and RAMHM = 0.
- XSRAM Space (0x0F_C000-0x0F_FFFF) when ROMHM = 1 and RAMHM = 1.

Table 3-19 shows the translation from the local memory map to the global memory map taking in consideration the different configurations of ROMHM and RAMHM.

**Table 3-19. ROMHM and RAMHM Address Location**

| Local Address | ROMHM | RAMHM | Global Address | Location |
|---|---|---|---|---|
| 0x4000 - 0x7FFF | 0 | X | 0x7F_4000 - 0x7F_7FFF | Internal Flash |
| | 1 | 0 | 0x14_4000 - 0x14_7FFF | External Space |
| | 1 | 1 | 0x0F_C000 - 0x0F_FFFF | Bottom of the Implemented RAM |
| 0x2000 - 0x3FFF | | | 0x0F_A000 - 0x0F_BFFF | Fixed up to 8K RAM |
| 0x2000 - 0x3FFF | 1 | 0 | 0x0F_E000 - 0x0F_FFFF | Fixed up to 8K RAM |

Table 3-20 describes the application note of the RAM configuration and its dedicated global address.

**Table 3-20. RAM Configuration**

| phase | RPAGE | ROMHM | RAMHM | RAM AREA | Global Address |
|---|---|---|---|---|---|
| After reset | RPAGE = 0xFD (Reset value) | 0 | 0 | 12 Kilobytes | 0x0F_D000 - 0x0F_FFFF |
| During setup | RPAGE = 0xFD (Reset value) | 1 | 1 | 24 Kilobytes | 0x0F_A000 - 0x0F_FFFF |
| Normal Operation | (0x00 <= RPAGE <= 0xF9) | 1 | 1 | 28 Kilobytes | 0x00_0000 - 0x0F_9FFF |
| | (0xFA <= RPAGE <= 0xFF) | 1 | 1 | 24 Kilobytes | 0x0F_A000 - 0x0F_FFFF |

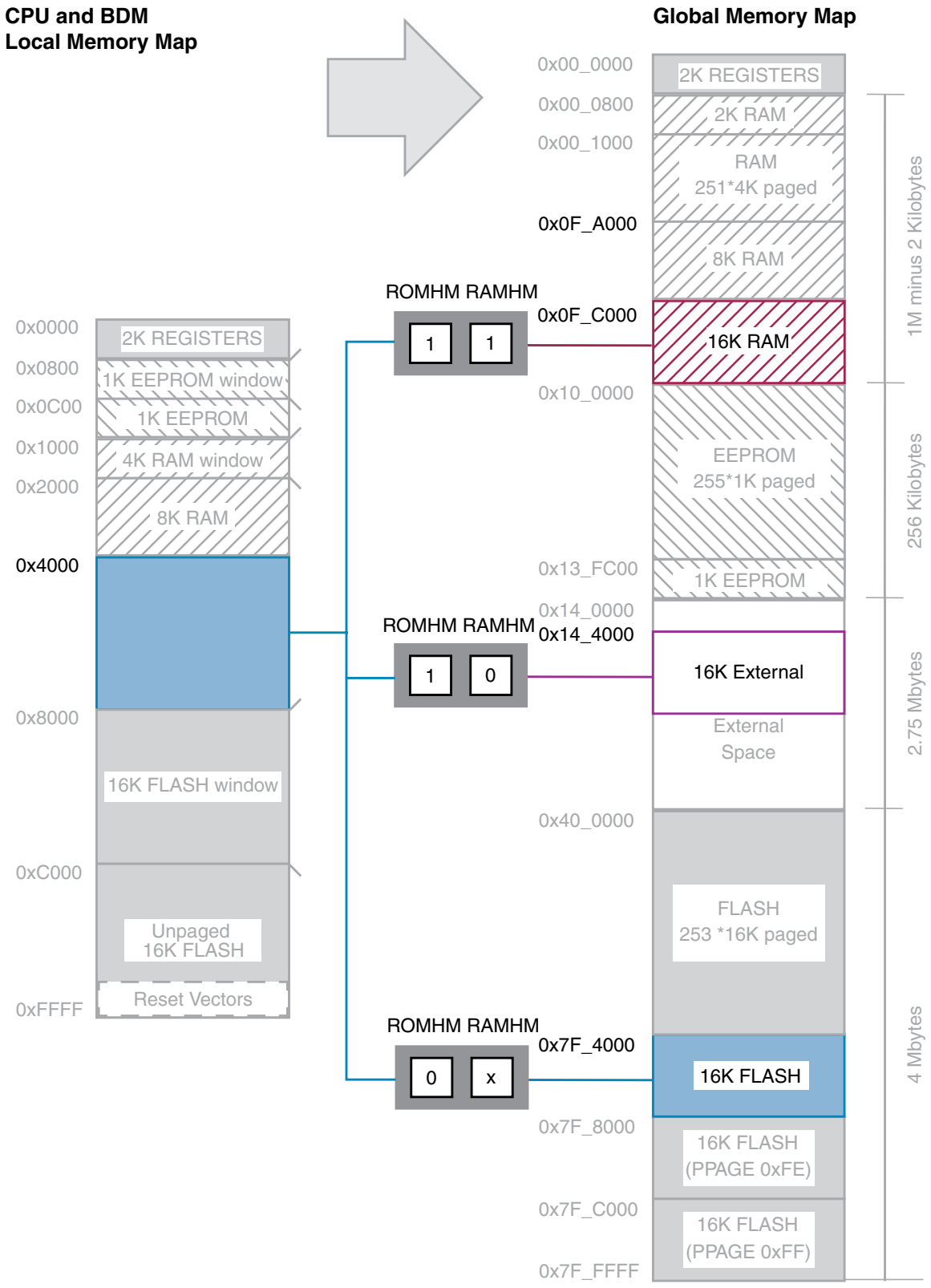



**Figure 3-21. ROMHM, RAMHM Memory Configuration**

#### 3.4.2.5.1 System XSRAM

System XSRAM has two ways to be accessed by the CPU. One is by the programming of RPAGE and the fixed XSRAM areas configured by the values of ROMHM, RAMHM, or by the usage of the global instruction and the usage of GPAGE.

Figure 3-22 shows the memory map for the implemented XSRAM. The size of the implemented XSRAM is done by the device definition and denoted by RAMSIZE.

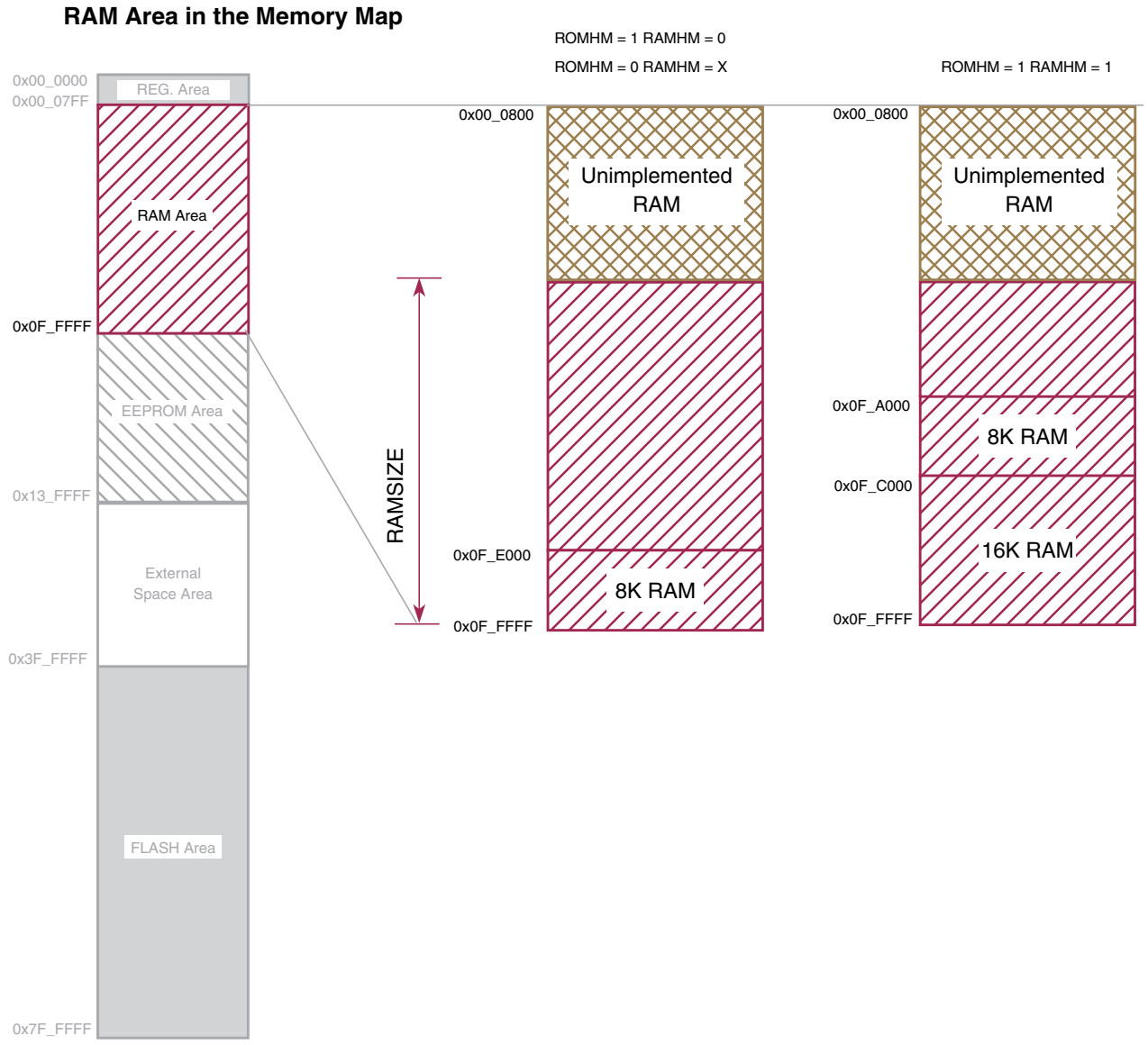



**Figure 3-22. S12XE System RAM in the Memory Map**

### 3.4.3 Chip Access Restrictions

CPU and XGATE accesses are watched in the memory protection unit (See MPU Block Guide). In case of access violation, the suspect master is acknowledged with an indication of an error; the victim target will not be accessed.

Other violations MPU is not handling are listed below.

#### 3.4.3.1 Illegal XGATE Accesses

A possible access error is flagged by the MMC and signalled to XGATE under the following conditions:

- XGATE performs misaligned word (in case of load-store or opcode or vector fetch accesses).
- XGATE accesses the register space (in case of opcode or vector fetch).
- XGATE performs a write to Flash in any modes (in case of load-store access).
- XGATE performs an access to a secured Flash in expanded modes (in case of load-store or opcode or vector fetch accesses).

For further details refer to the XGATE Block Guide.

### 3.4.4 Chip Bus Control

The MMC controls the address buses and the data buses that interface the S12X masters (CPU, BDM and XGATE) with the rest of the system (master buses). In addition the MMC handles all CPU read data bus swapping operations. All internal and external resources are connected to specific target buses (see Figure 3-23[1]).

1. Doted blocks and lines are optional. Please refer to the Device User Guide for their availlibilities.

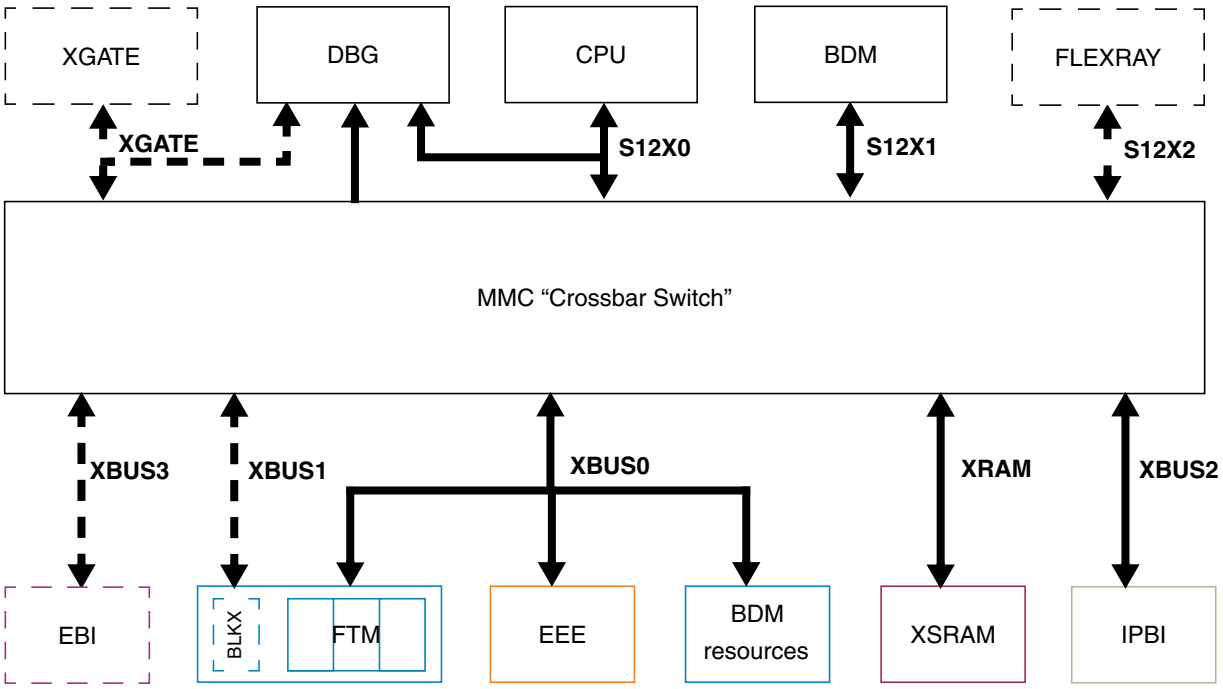



**Figure 3-23. MMC Block Diagram**

### 3.4.4.1 Master Bus Prioritization regarding access conflicts on Target Buses

The arbitration scheme allows only one master to be connected to a target at any given time. The following rules apply when prioritizing accesses from different masters to the same target bus:

- CPU always has priority over BDM and XGATE.
- XGATE access to PRU registers constitutes a special case. It is always granted and stalls the CPU for its duration.
- XGATE has priority over BDM.
- BDM has priority over CPU and XGATE when its access is stalled for more than 128 cycles. In the later case the suspect master will be stalled after finishing the current operation and the BDM will gain access to the bus.
- In emulation modes all internal accesses are visible on the external bus as well and the external bus is used during access to the PRU registers.

## 3.5 Initialization/Application Information

### 3.5.1 CALL and RTC Instructions

CALL and RTC instructions are uninterruptable CPU instructions that automate page switching in the program page window. The CALL instruction is similar to the JSR instruction, but the subroutine that is

called can be located anywhere in the local address space or in any Flash or ROM page visible through the program page window. The CALL instruction calculates and stacks a return address, stacks the current PPAGE value and writes a new instruction-supplied value to the PPAGE register. The PPAGE value controls which of the 256 possible pages is visible through the 16 Kbyte program page window in the 64 Kbyte local CPU memory map. Execution then begins at the address of the called subroutine.

During the execution of the CALL instruction, the CPU performs the following steps:

1. Writes the current PPAGE value into an internal temporary register and writes the new instruction-supplied PPAGE value into the PPAGE register
2. Calculates the address of the next instruction after the CALL instruction (the return address) and pushes this 16-bit value onto the stack
3. Pushes the temporarily stored PPAGE value onto the stack
4. Calculates the effective address of the subroutine, refills the queue and begins execution at the new address

This sequence is uninterruptable. There is no need to inhibit interrupts during the CALL instruction execution. A CALL instruction can be performed from any address to any other address in the local CPU memory space.

The PPAGE value supplied by the instruction is part of the effective address of the CPU. For all addressing mode variations (except indexed-indirect modes) the new page value is provided by an immediate operand in the instruction. In indexed-indirect variations of the CALL instruction a pointer specifies memory locations where the new page value and the address of the called subroutine are stored. Using indirect addressing for both the new page value and the address within the page allows usage of values calculated at run time rather than immediate values that must be known at the time of assembly.

The RTC instruction terminates subroutines invoked by a CALL instruction. The RTC instruction unstacks the PPAGE value and the return address and refills the queue. Execution resumes with the next instruction after the CALL instruction.

During the execution of an RTC instruction the CPU performs the following steps:

1. Pulls the previously stored PPAGE value from the stack
2. Pulls the 16-bit return address from the stack and loads it into the PC
3. Writes the PPAGE value into the PPAGE register
4. Refills the queue and resumes execution at the return address

This sequence is uninterruptable. The RTC can be executed from anywhere in the local CPU memory space.

The CALL and RTC instructions behave like JSR and RTS instruction, they however require more execution cycles. Usage of JSR/RTS instructions is therefore recommended when possible and CALL/RTC instructions should only be used when needed. The JSR and RTS instructions can be used to access subroutines that are already present in the local CPU memory map (i.e. in the same page in the program memory page window for example). However calling a function located in a different page requires usage of the CALL instruction. The function must be terminated by the RTC instruction. Because the RTC instruction restores contents of the PPAGE register from the stack, functions terminated with the RTC instruction must be called using the CALL instruction even when the correct page is already present

in the memory map. This is to make sure that the correct PPAGE value will be present on stack at the time of the RTC instruction execution.

### 3.5.2 Port Replacement Registers (PRRs)

Registers used for emulation purposes must be rebuilt by the in-circuit emulator hardware to achieve full emulation of single chip mode operation. These registers are called port replacement registers (PRRs) (see Table 1-25). PRRs are accessible from CPU, BDM and XGATE using different access types (word aligned, word-misaligned and byte).

Each access to PRRs will be extended to 2 bus cycles for write or read accesses independent of the operating mode. In emulation modes all write operations result in simultaneous writing to the internal registers (peripheral access) and to the emulated registers (external access) located in the PRU in the emulator. All read operations are performed from external registers (external access) in emulation modes. In all other modes the read operations are performed from the internal registers (peripheral access).

Due to internal visibility of CPU accesses the CPU will be halted during XGATE or BDM access to any PRR. This rule applies also in normal modes to ensure that operation of the device is the same as in emulation modes.

A summary of PRR accesses:

- An aligned word access to a PRR will take 2 bus cycles.
- A misaligned word access to a PRRs will take 4 cycles. If one of the two bytes accessed by the misaligned word access is not a PRR, the access will take only 3 cycles.
- A byte access to a PRR will take 2 cycles.

**Table 3-21. PRR Listing**

| PRR Name | PRR Local Address | PRR Location |
|---|---|---|
| PORTA | 0x0000 | PIM |
| PORTB | 0x0001 | PIM |
| DDRA | 0x0002 | PIM |
| DDRB | 0x0003 | PIM |
| PORTC | 0x0004 | PIM |
| PORTD | 0x0005 | PIM |
| DDRC | 0x0006 | PIM |
| DDRD | 0x0007 | PIM |
| PORTE | 0x0008 | PIM |
| DDRE | 0x0009 | PIM |
| MMCCTL0 | 0x000A | MMC |
| MODE | 0x000B | MMC |
| PUCR | 0x000C | PIM |
| RDRIV | 0x000D | PIM |
| EBICTL0 | 0x000E | EBI |
| EBICTL1 | 0x000F | EBI |
| Reserved | 0x0012 | MMC |
| MMCCTL1 | 0x0013 | MMC |
| ECLKCTL | 0x001C | PIM |
| Reserved | 0x001D | PIM |
| PORTK | 0x0032 | PIM |
| DDRK | 0x0033 | PIM |

## 3.5.3 On-Chip ROM Control

The MCU offers two modes to support emulation. In the first mode (called generator) the emulator provides the data instead of the internal FLASH and traces the CPU actions. In the other mode (called observer) the internal FLASH provides the data and all internal actions are made visible to the emulator.

### 3.5.3.1 ROM Control in Single-Chip Modes

In single-chip modes the MCU has no external bus. All memory accesses and program fetches are internal (see Figure 3-24).

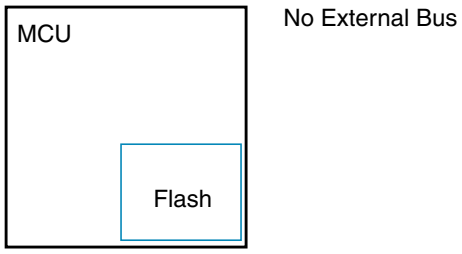



**Figure 3-24. ROM in Single Chip Modes**

### 3.5.3.2 ROM Control in Emulation Single-Chip Mode

In emulation single-chip mode the external bus is connected to the emulator. If the EROMON bit is set, the internal FLASH provides the data and the emulator can observe all internal CPU actions on the external bus. If the EROMON bit is cleared, the emulator provides the data (generator) and traces the all CPU actions (see Figure 3-25).

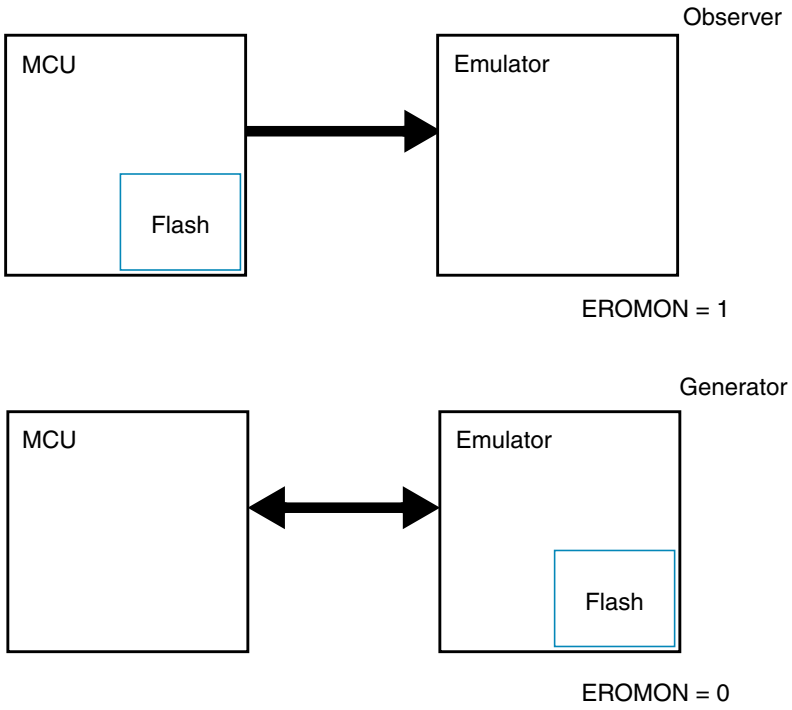



**Figure 3-25. ROM in Emulation Single-Chip Mode**

### 3.5.3.3 ROM Control in Normal Expanded Mode

In normal expanded mode the external bus will be connected to the application. If the ROMON bit is set, the internal FLASH provides the data. If the ROMON bit is cleared, the application memory provides the data (see Figure 3-26).

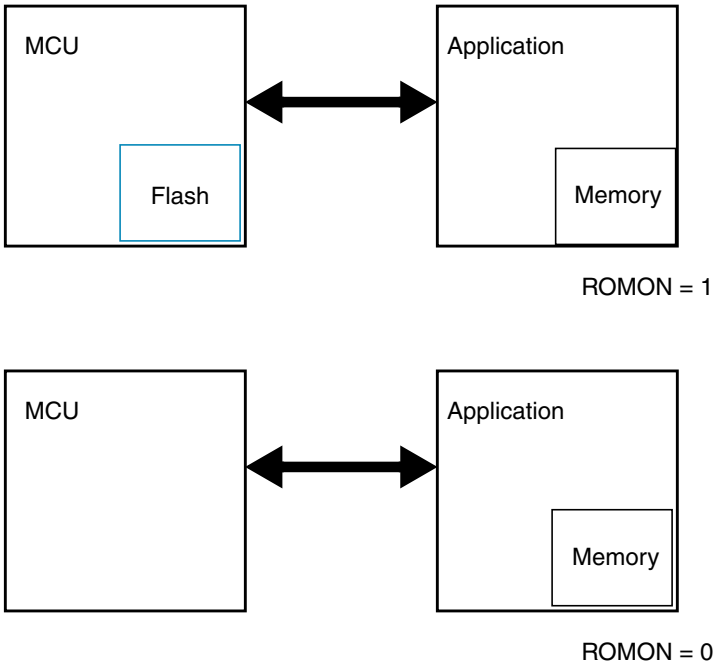



**Figure 3-26. ROM in Normal Expanded Mode**

### 3.5.3.4 ROM Control in Emulation Expanded Mode

In emulation expanded mode the external bus will be connected to the emulator and to the application. If the ROMON bit is set, the internal FLASH provides the data. If the EROMON bit is set as well the emulator observes all CPU internal actions, otherwise the emulator provides the data and traces all CPU actions (see Figure 3-27). When the ROMON bit is cleared, the application memory provides the data and the emulator will observe the CPU internal actions (see Figure 3-28).

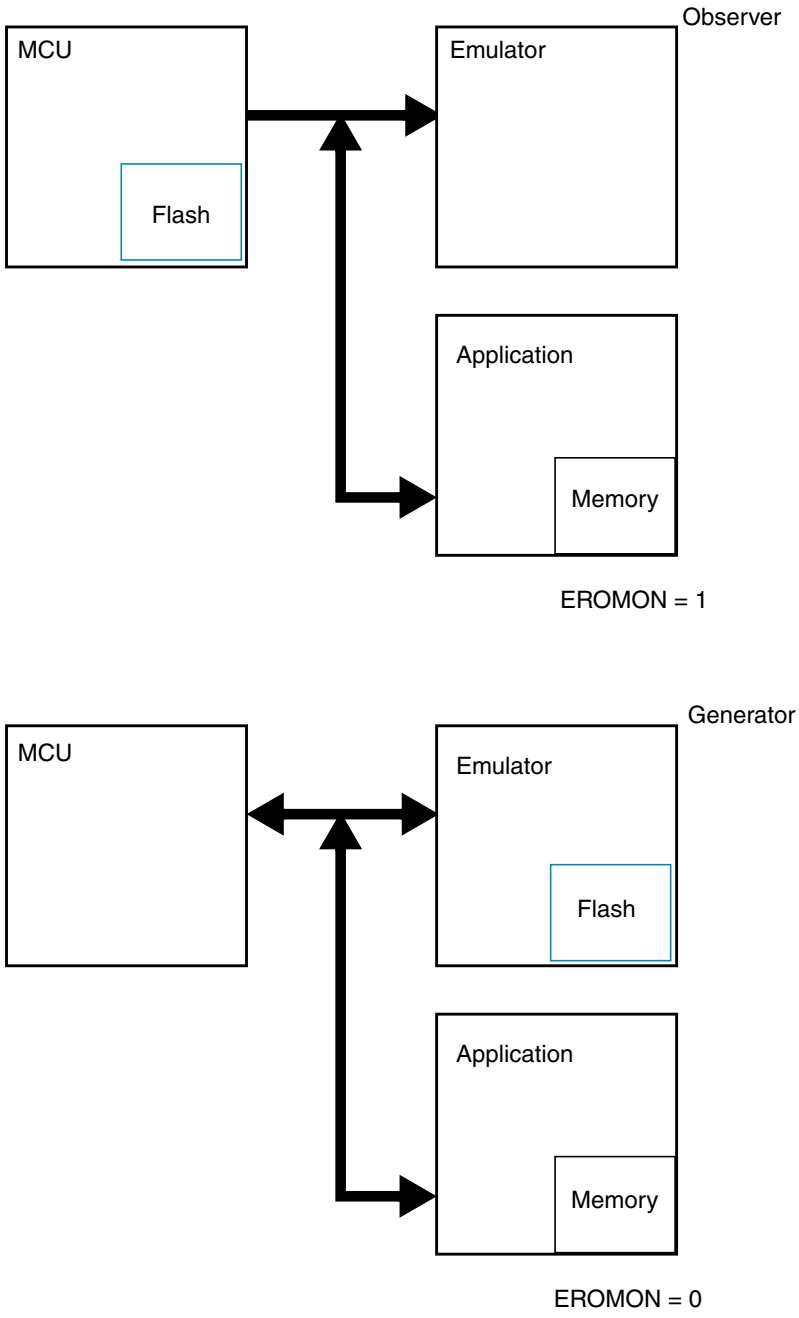



**Figure 3-27. ROMON = 1 in Emulation Expanded Mode**

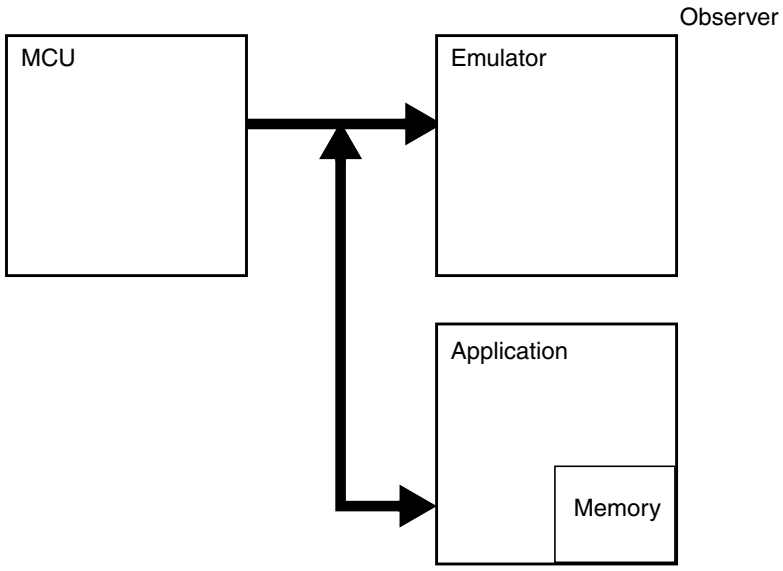



**Figure 3-28. ROMON = 0 in Emulation Expanded Mode**

### 3.5.3.5 ROM Control in Special Test Mode

In special test mode the external bus is connected to the application. If the ROMON bit is set, the internal FLASH provides the data, otherwise the application memory provides the data (see Figure 3-29).

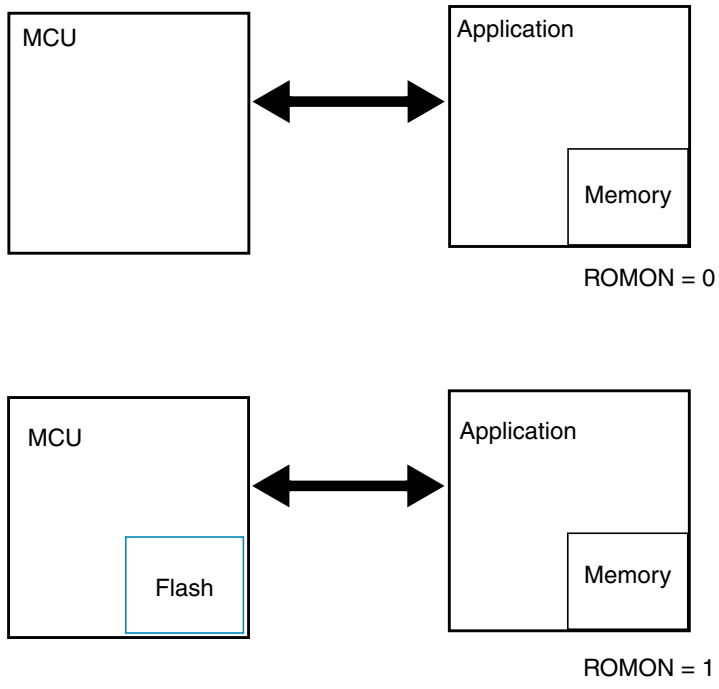



**Figure 3-29. ROM in Special Test Mode**

# Chapter 4
# Memory Protection Unit (S12XMPUV1)

**Table 4-1. Revision History**

| Revision Number | Revision Date | Sections Affected | Description of Changes |
|---|---|---|---|
| V01.04 | 14 Sep 2005 | 4.3.1.1/4-231<br>4.4.1/4-237 | - Added note to only use the CPU to clear the AE flag.<br>- Added disclaimer to avoid changing descriptors while they are in use because of other bus-masters doing accesses. |
| V01.05 | 14 Mar 2006 | 4.3.1.1/4-231<br>4.4/4-237 | - Clarified that interrupt generation is independent of AEF bit state.<br>- Corrected preliminary statement about execution of violating accesses. |
| V01.06 | 09 Oct 2006 | | - Made Revision History entries public. |

## 4.1 Introduction

The MPU module provides basic functionality required to protect memory mapped resources from undesired accesses. Multiple address range comparators compare memory accesses against eight memory protection descriptors located in the MPU module to determine if each access is valid or not. The comparison is sensitive to which bus master generates the access and the type of the access.

The MPU module can be used to isolate memory ranges accessible by different bus masters. It can be also be used by an operating system or software kernel to isolate the regions of memory “legally” available to specific software tasks, with the kernel re-configuring the task specific memory protection descriptors in supervisor state during task-switching.

### 4.1.1 Preface

The following terms and abbreviations are used in the document.

**Table 4-2. Terminology**

| Term | Meaning |
|---|---|
| MCU | Micro-Controller Unit |
| MPU | Memory Protection Unit |
| CPU | S12X Central Processing Unit (see S12XCPU Reference Manual) |
| XGATE | XGATE Co-processor (see XGATE chapter) |
| supervisor state | refers to the supervisor state of the S12XCPU (see S12XCPU Reference Manual) |
| user state | refers to the user state of the S12XCPU (see S12XCPU Reference Manual) |

### 4.1.2 Overview

The MPU module monitors the bus activity of each bus master. The data describing each access is fed into multiple address range comparators. The output of the comparators is used to determine if a particular

access is allowed or represents an access violation. If an access violation caused by the S12X CPU is detected, the MPU module raises an access violation interrupt. If the MPU module detects an access violation caused by a bus master other than the S12X CPU, it flags an access error condition to the respective master. In addition to the restrictions defined for memory ranges in the MPU descriptors, accesses to memory not covered by any MPU descriptor (even read accesses!) are considered access violations.

Figure 4-1 shows a block diagram of the MPU module.

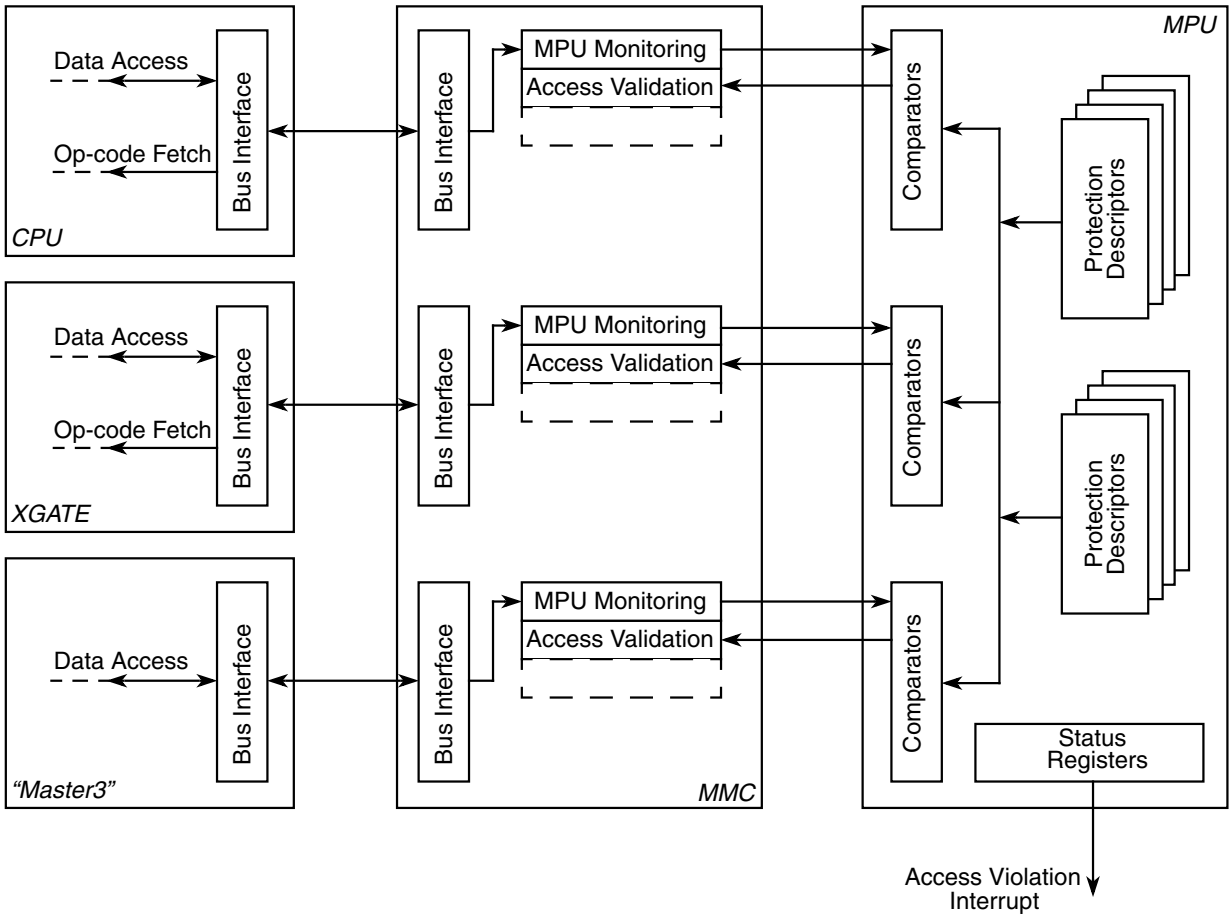



**Figure 4-1. Block Diagram**

## 4.1.3 Features

- Protects memory from undesired accesses coming from up to 3 bus masters[1]
- Eight memory protection descriptors
  - — each descriptor can cover the full global memory map (8 MBytes)
  - — each descriptor has a granularity of 8 Bytes

1. Master 3 can be implemented or left out depending the chip configuration. Please refer to the Device Reference Manual for information about the availability and function of Master 3.

- Each descriptor can be configured to allow one of four types of access privilege for the defined memory region
  - — Bus master has full access (read, write and execute enabled)
  - — Bus master can read and execute (write illegal)
  - — Bus master can read and write (execution illegal)
  - — Bus master can only read (write and execution illegal)
- Accesses to memory not covered by any protection descriptor will cause an access violation

### 4.1.4 Modes of Operation

The MPU module can be used in all MCU modes.

## 4.2 External Signal Description

The MPU module has no external signals.

## 4.3 Memory Map and Register Definition

This section provides a detailed description of address space and registers used by the MPU module.

## 4.3.1 Register Descriptions

This section describes in address order all the MPU module registers and their individual bits.

| Register Name | | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|---|
| 0x0000 MPUFLG | R | AEF | WPF | NEXF | 0 | 0 | 0 | 0 | SVSF |
| | W | | | | | | | | |
| 0x0001 MPUASTAT0 | R | 0 | ADDR[22:16] | | | | | | |
| | W | | | | | | | | |
| 0x0002 MPUASTAT1 | R | ADDR[15:8] | | | | | | | |
| | W | | | | | | | | |
| 0x0003 MPUASTAT2 | R | ADDR[7:0] | | | | | | | |
| | W | | | | | | | | |
| 0x0004 Reserved | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | W | | | | | | | | |
| 0x0005 MPUSEL | R | SVSEN | 0 | 0 | 0 | 0 | SEL[2:0] | | |
| | W | | | | | | | | |
| 0x0006 MPUDESC0(1) | R | MSTR0 | MSTR1 | MSTR2 | MSTR3 | LOW_ADDR[22:19] | | | |
| | W | | | | | | | | |
| 0x0007 MPUDESC1[1] | R | LOW_ADDR[18:11] | | | | | | | |
| | W | | | | | | | | |
| 0x0008 MPUDESC2[1] | R | LOW_ADDR[10:3] | | | | | | | |
| | W | | | | | | | | |
| 0x0009 MPUDESC3[1] | R | WP | NEX | 0 | 0 | HIGH_ADDR[22:19] | | | |
| | W | | | | | | | | |
| 0x000A MPUDESC4[1] | R | HIGH_ADDR[18:11] | | | | | | | |
| | W | | | | | | | | |
| 0x000B MPUDESC5[1] | R | HIGH_ADDR[10:3] | | | | | | | |
| | W | | | | | | | | |

= Unimplemented or Reserved

1. The module addresses 0x0006–0x000B represent a window in the register map through which different descriptor registers are visible.

**Figure 4-2. MPU Register Summary**

### 4.3.1.1 MPU Flag Register (MPUFLG)

Address: Module Base + 0x0000

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | AEF | WPF | NEXF | 0 | 0 | 0 | 0 | SVSF |
| W | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 4-3. MPU Flag Register (MPUFLG)**

Read: Anytime

Write: Write of 1 clears flag, write of 0 ignored

**Table 4-3. MPUFLG Field Descriptions**

| Field | Description |
|---|---|
| 7<br>AEF | **Access Error Flag** — This bit is the CPU access error interrupt flag. It is set if a CPU access violation has occurred. At the same time this bit is set, all the other status flags in this register and the access violation address bits in the MPUASTATn registers are captured. Clear this flag by writing a one.<br>**Note:** If a CPU access error is flagged and both the WPF bit and the NEXF bit are zero, the access violation was caused by an access to memory not covered by the MPU descriptors.<br>**Note:** While this bit is set, the CPU in supervisor state (“Master 0”) can read from and write to the peripheral register space even if there is no memory protection descriptor explicitly allowing this. This is to prevent the case that the CPU cannot clear the AEF bit if the registers are write protected for the CPU in supervisor state.<br>**Note:** This bit should only be cleared by an access from the S12X CPU. Otherwise, when using one of the other masters (such as the XGATE) to clear this bit, the status flags and the address status registers may not get updated correctly if a CPU access causes a violation in the same bus cycle. |
| 6<br>WPF | **Write-Protect Violation Flag** — This flag is set if the current CPU access violation has occurred because of an attempt to write to memory configured as read-only. The WPF bit is read-only; it will be automatically updated when the next access violation is flagged with the AEF bit. |
| 5<br>NEXF | **No-Execute Violation Flag** — This bit is set if the current CPU access violation has occurred because of an attempt to fetch code from memory configured as No-Execute. The NEXF bit is read-only; it will be automatically updated when the next access violation is flagged with the AEF bit. |
| 0<br>SVSF | **Supervisor State Flag** — This bit is set if the current CPU access violation occurred while the CPU was in supervisor state. This bit is cleared if the current CPU access violation occurred while the CPU was in user state. The supervisor state flag is read-only; it will be automatically updated when the next CPU access violation is flagged with the AEF bit. |

If the AEF bit is set further violations are not captured into the MPU status registers. The status of the AEF bit has no effect on the access restrictions, i.e. access restrictions for all masters are still enforced if the AEF bit is set. Also, the non-maskable hardware interrupt for violating accesses coming from the S12X CPU is generated regardless of the state of the AEF bit.

### 4.3.1.2 MPU Address Status Register 0 (MPUASTAT0)

Address: Module Base + 0x0001

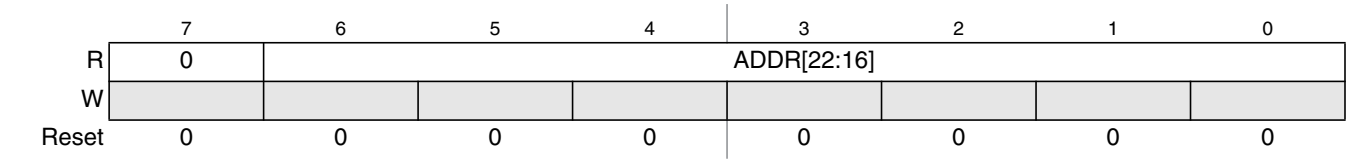



**Figure 4-4. MPU Address Status Register 0 (MPUASTAT0)**

Read: Anytime

Write: Never

**Table 4-4. MPUASTAT0 Field Descriptions**

| Field | Description |
|---|---|
| 6–0<br>ADDR[22:16] | **Access violation address bits** — The ADDR[22:16] bits contain bits [22:16] of the global address which caused the current access violation interrupt. These bits are undefined if the access error flag bit (AEF) in the MPUFLG register is not set. |

### 4.3.1.3 MPU Address Status Register 1 (MPUASTAT1)

Address: Module Base + 0x0002

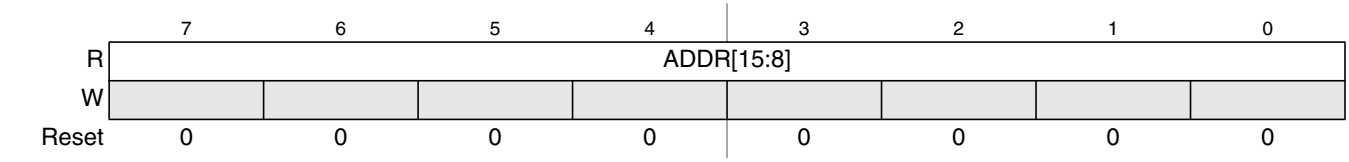



**Figure 4-5. MPU Address Status Register 1 (MPUASTAT1)**

Read: Anytime

Write: Never

**Table 4-5. MPUASTAT1 Field Descriptions**

| Field | Description |
|---|---|
| 7–0<br>ADDR[15:8] | **Access violation address bits** — The ADDR[15:8] bits contain bits [15:8] of the global address which caused the current access violation interrupt. These bits are undefined if the access error flag bit (AEF) in the MPUFLG register is not set. |

### 4.3.1.4 MPU Address Status Register 2 (MPUASTAT2)

Address: Module Base + 0x0003

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | ADDR[7:0] | | | | | | | |
| W | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 4-6. MPU Address Status Register (MPUASTAT2)**

Read: Anytime

Write: Never

**Table 4-6. MPUASTAT2 Field Descriptions**

| Field | Description |
|---|---|
| 7–0<br>ADDR[7:0] | **Access violation address bits** — The ADDR[7:0] bits contain bits [7:0] of the global address which caused the current access violation interrupt. These bits are undefined if the access error flag bit (AEF) in the MPUFLG register is not set. |

### 4.3.1.5 MPU Descriptor Select Register (MPUSEL)

Address: Module Base + 0x0005

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | SVSEN | 0 | 0 | 0 | 0 | SEL[2:0] | | |
| W | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 4-7. MPU Descriptor Select Register (MPUSEL)**

Read: Anytime

Write: Anytime

**Table 4-7. MPUSEL Field Descriptions**

| Field | Description |
|---|---|
| 7<br>SVSEN | **MPU supervisor state enable bit** — This bit enables the memory protection for the CPU in supervisor state. If this bit is cleared, the MPU does not affect any accesses coming from the CPU in supervisor state. This is to prevent the CPU from locking out itself while configuring the protection descriptors (during initialization after a system reset and during the update of the protection descriptors for a task switch). The memory protection functionality for the other bus-masters is unaffected by this bit.<br>0 MPU is disabled for the CPU in supervisor state<br>1 MPU is enabled for the CPU in supervisor state |
| 2–0<br>SEL[2:0] | **Descriptor select bits** — The SEL[2:0] bits select which descriptor is visible in the MPU Descriptor Register window (MPUDESC0—MPUDESC5). |

### 4.3.1.6 MPU Descriptor Register 0 (MPUDESC0)

Address: Module Base + 0x0006

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | MSTR0 | MSTR1 | MSTR2 | MSTR3 | LOW_ADDR[22:19] | | | |
| Reset | 1[1] | 1[1] | 1[1] | 1[2] | 0 | 0 | 0 | 0 |

1. initialized as set for descriptor 0 only, cleared for all others
2. initialized as set for descriptor 0 only, if MSTR3 is implemented on the device

**Figure 4-8. MPU Descriptor Register 0 (MPUDESC0)**

Read: Anytime

Write: Anytime

**Table 4-8. MPUDESC0 Field Descriptions**

| Field | Description |
|---|---|
| 7<br>MSTR0 | **Master 0 select bit** — If this bit is set the descriptor is valid for bus master 0 (CPU in supervisor state). |
| 6<br>MSTR1 | **Master 1 select bit** — If this bit is set the descriptor is valid for bus master 1 (CPU in user state). |
| 5<br>MSTR2 | **Master 2 select bit** — If this bit is set the descriptor is valid for bus master 2 (XGATE). |
| 4<br>MSTR3 | **Master 3 select bit** — If this bit is set the descriptor is valid for bus master 3. |
| 3–0<br>LOW_ADDR[22:19] | **Memory range lower boundary address bits** — The LOW_ADDR[22:19] bits represent bits [22:19] of the global memory address that is used as the lower boundary for the described memory range. |

A descriptor can be configured as valid for more than one bus-master at the same time by setting multiple Master select bits to one. Setting all Master select bits of a descriptor to zero disables the descriptor.

### 4.3.1.7 MPU Descriptor Register 1 (MPUDESC1)

Address: Module Base + 0x0007

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | LOW_ADDR[18:11] | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 4-9. MPU Descriptor Register 1 (MPUDESC1)**

Read: Anytime

Write: Anytime

**Table 4-9. MPUDESC1 Field Descriptions**

| Field | Description |
|---|---|
| 7–0<br>LOW_ADDR[18:11] | **Memory range lower boundary address bits** — The LOW_ADDR[18:11] bits represent bits [18:11] of the global memory address that is used as the lower boundary for the described memory range. |

### 4.3.1.8 MPU Descriptor Register 2 (MPUDESC2)

Address: Module Base + 0x0008

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | LOW_ADDR[10:3] | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 4-10. MPU Descriptor Register 2 (MPUDESC2)**

Read: Anytime

Write: Anytime

**Table 4-10. MPUDESC2 Field Descriptions**

| Field | Description |
|---|---|
| 7–0<br>LOW_ADDR[10:3] | **Memory range lower boundary address bits** — The LOW_ADDR[10:3] bits represent bits [10:3] of the global memory address that is used as the lower boundary for the described memory range. |

### 4.3.1.9 MPU Descriptor Register 3 (MPUDESC3)

Address: Module Base + 0x0009

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | WP | NEX | 0 | 0 | HIGH_ADDR[22:19] | | | |
| W | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 |

**Figure 4-11. MPU Descriptor Register 3 (MPUDESC3)**

Read: Anytime

Write: Anytime

**Table 4-11. MPUDESC3 Field Descriptions**

| Field | Description |
|---|---|
| 7<br>WP | **Write-Protect bit** — The WP bit causes the described memory range to be treated as write-protected. If this bit is set every attempt to write in the described memory range causes an access violation. |

| Field | Description |
|---|---|
| 6<br>NEX | **No-Execute bit** — The NEX bit prevents the described memory range from being used as code memory. If this bit is set every Op-code fetch in this memory range causes an access violation. |
| 3–0<br>HIGH_ADDR[22:19] | **Memory range upper boundary address bits** — The HIGH_ADDR[22:19] bits represent bits [22:19] of the global memory address that is used as the upper boundary for the described memory range. |

### 4.3.1.10 MPU Descriptor Register 4 (MPUDESC4)

Address: Module Base + 0x000A

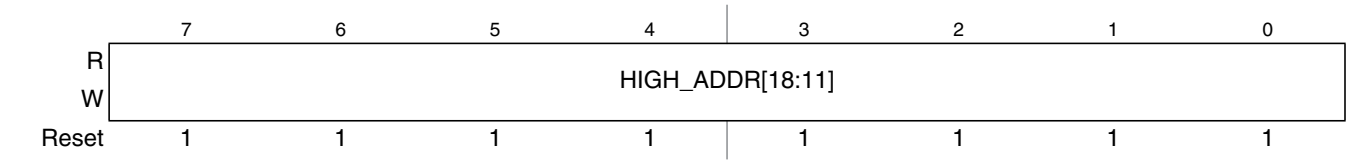



**Figure 4-12. MPU Descriptor Register 4 (MPUDESC4)**

Read: Anytime

Write: Anytime

**Table 4-12. MPUDESC4 Field Descriptions**

| Field | Description |
|---|---|
| 7–0<br>HIGH_ADDR[18:11] | **Memory range upper boundary address bits** — The HIGH_ADDR[18:11] bits represent bits [18:11] of the global memory address that is used as the upper boundary for the described memory range. |

### 4.3.1.11 MPU Descriptor Register 5 (MPUDESC5)

Address: Module Base + 0x000B

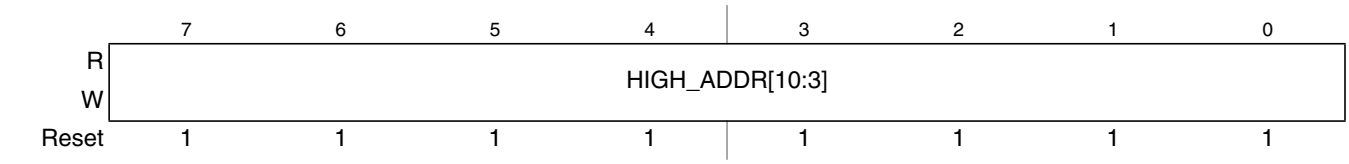



**Figure 4-13. MPU Descriptor Register 5 (MPUDESC5)**

Read: Anytime

Write: Anytime

**Table 4-13. MPUDESC5 Field Descriptions**

| Field | Description |
|---|---|
| 7–0<br>HIGH_ADDR[10:3] | **Memory range upper boundary address bits** — The HIGH_ADDR[10:3] bits represent bits [10:3] of the global memory address that is used as the upper boundary for the described memory range. |

## 4.4 Functional Description

The MPU module provides memory protection for accesses coming from multiple masters in the system. This is done by monitoring bus traffic of each master and compare this with the configuration information from a set of eight programmable descriptors located in the MPU module. If the MPU module detects an access violation caused by the S12X CPU, it will assert the CPU access violation interrupt signal. If the MPU module detects an access violation caused by a bus master other than the S12X CPU, it raises an access error signal. Please refer to the documentation chapter of the individual master modules (i.e. XGATE, etc.) for more information about the access error condition.

Violating accesses are not executed. The return value of a violating read access is undefined for both 8 bit and 16 bit accesses.

**NOTE**

Accesses from BDM are not restricted. BDM hardware accesses always bypass the MPU module. During execution of BDM firmware code S12X CPU accesses are masked from the MPU module as well.

### 4.4.1 Protection Descriptors

Each of the eight protection descriptors can be used to restrict the allowed types of memory accesses for a given memory range. Each of these memory ranges can cover up the entire 23 bits global memory range (8 MBytes).

The descriptors are banked in the MPU module register map.

Each descriptor can be selected for modifying using the SEL bits in the MPU Descriptor Select (MPUSEL) register.

Table 4-14 gives an overview of the types of accesses that can be configured using the protection descriptors.

**Table 4-14. Access Types**

| WP | NEX | Meaning |
|---|---|---|
| 0 | 0 | read, write and execute |
| 0 | 1 | read, write |
| 1 | 0 | read and execute |
| 1 | 1 | read only |

The granularity of each descriptor is 8 bytes. This means the protection comparators in the MPU module cover only address bits [22:3] of each access. The lower address bits [2:0] are ignored.

**NOTE**

A mis-aligned word access to the upper boundary address of a descriptor is always flagged as an access violation.

**NOTE**

Configuring the lower boundary address of a descriptor to be higher than the upper boundary address of a descriptor causes this descriptor to be ignored by the comparator block. This effectively disables the descriptor.

**NOTE**

Avoid changing descriptors while they are in active use to validate accesses from bus-masters. This can be done by temporarily disabling the affected master during the update (XGATE, Master 3, switch S12X CPU states). Otherwise accesses from bus-masters affected by a descriptor which is updated concurrently could yield undefined results.

#### 4.4.1.1 Overlapping Descriptors

If the memory ranges of two protection descriptors defined for the same bus-master overlap, the access restrictions for the overlapped memory range are accumulated. For example:

- a memory protection descriptor defines memory range 0x40_0000–0x41_FFFF as WP=1, NEX=0 (read and execute)
- another descriptor defines memory range 0x41_0000–0x43_FFFF as WP=0, NEX=1 (read and write)
- the resulting access rights for the overlapping range 0x41_0000–0x41_FFFF are WP=1, NEX=1 (read only)

#### 4.4.1.2 Implicitly defined memory descriptors

As mentioned in the bit description of the Access Error Flag (AEF) in the MPUFLG register (Table 4-3), there is an additional memory range implicitly defined only while the AEF bit is set: The CPU in supervisor state can read from and write to the peripheral register space even if there is no memory protection descriptor explicitly allowing this. This is to prevent the case that the CPU cannot clear the AEF bit if the registers are write protected for the CPU in supervisor state.

The register address space containing the PAGE registers (EPAGE, RPAGE, GPAGE, PPAGE) at 0x0010–0x0017 gets special treatment. It is defined like this:

- The S12X CPU can always read and write these registers, regardless of the configuration in the descriptors.
- XGATE or Master3 (if available) are never allowed to read or write these registers, even if the descriptor configuration allows accesses for other masters than the S12X CPU.

#### 4.4.1.3 Op-code pre-fetch cycles and the NEX bit

Some bus-masters (CPU, XGATE) do a pre-fetch of program-code past the current instruction. The S12XCPU pre-fetches two words past the current instruction, the XGATE pre-fetches one word, even if the pre-fetched code is not executed. The MPU module has no way of knowing this at the time when the pre-fetch cycles occur. Therefore this will result in an access violation if the op-code pre-fetch accesses a memory range marked as “No-Execute” (NEX=1). This must be taken into account when defining memory

ranges with the NEX bit set adjacent to memory used for program code. The best way to do this would be to leave some fill-bytes between the memory ranges in this case, i.e. do not set the upper memory boundary to the address of the last op-code but to a following address which is at least two words (four bytes) away.

### 4.4.2 Interrupts

This section describes all interrupts originated by the MPU module.

#### 4.4.2.1 Description of Interrupt Operation

The MPU module generates one interrupt request. It cannot be masked locally in the MPU module and is meant to be used as the source of a non-maskable hardware interrupt request for the S12X CPU

**Table 4-15. Interrupt vectors**

| Interrupt Source | CCR Mask | Local Enable |
|---|---|---|
| S12X CPU access error interrupt (AEF) | – | – |

#### 4.4.2.2 CPU Access Error Interrupt

An S12X CPU access error interrupt request is generated if the MPU module has detected an illegal memory access originating from the S12X CPU. This is a non-maskable hardware interrupt. Due to the non-maskable nature of this interrupt, the de-assertion of this interrupt request is coupled to the S12X CPU interrupt vector fetch instead of the local access error flag (AEF). This means leaving the access error flag (AEF) in the MPUFLG register set will not cause the same interrupt to be serviced again after leaving the interrupt service routine with “RTI”. Instead, the interrupt request will be asserted again only when the next illegal S12X CPU access is detected.

## 4.5 Initialization/Application Information

### 4.5.1 Initialization

After reset the MPU module is in an unconfigured state, with all eight protection descriptors covering the whole memory map. The master bits are all set for descriptor “0” and cleared for all other descriptors. The S12XCPU in supervisor state can access everything because the SVSEN bit in the MPUSEL register is cleared by a system reset. After system reset every master has full access to the memory map because of descriptor “0”.

In order to use the MPU module to protect memory ranges from undesired accesses, software needs to:

- Initialize the protection descriptors.
- Make sure there are meaningful interrupt service routines defined for the Access Violation interrupts because these are non-maskable (See S12XINT chapter for details).
- Initialize peripherals and other masters for use (i.e. set-up XGATE, Master3 if applicable).
- Enable the MPU protection for the S12X CPU in supervisor state, if desired.
- Switch the S12X CPU to user state, if desired.

# Chapter 5
# External Bus Interface (S12XEBIV4)

**Table 5-1. Revision History**

| Revision Number | Revision Date | Sections Affected | Description of Changes |
|---|---|---|---|
| V04.01 | 12 Sep 2005 | | - Added CSx stretch description. |
| V04.02 | 23 May 2006 | | - Internal updates |
| V04.03 | 24 Jul 2006 | | - Removed term IVIS |

## 5.1 Introduction

This document describes the functionality of the XEBI block controlling the external bus interface.

The XEBI controls the functionality of a non-multiplexed external bus (a.k.a. ‘expansion bus’) in relationship with the chip operation modes. Dependent on the mode, the external bus can be used for data exchange with external memory, peripherals or PRU, and provide visibility to the internal bus externally in combination with an emulator.

### 5.1.1 Glossary or Terms

| | |
|---|---|
| bus clock | System Clock. Refer to CRG Block Guide. |
| expanded modes | Normal Expanded Mode<br>Emulation Single-Chip Mode<br>Emulation Expanded Mode<br>Special Test Mode |
| single-chip modes | Normal Single-Chip Mode<br>Special Single-Chip Mode |
| emulation modes | Emulation Single-Chip Mode<br>Emulation Expanded Mode |
| normal modes | Normal Single-Chip Mode<br>Normal Expanded Mode |
| special modes | Special Single-Chip Mode<br>Special Test Mode |
| NS | Normal Single-Chip Mode |
| SS | Special Single-Chip Mode |
| NX | Normal Expanded Mode |
| ES | Emulation Single-Chip Mode |
| EX | Emulation Expanded Mode |
| ST | Special Test Mode |
| external resource | Addresses outside MCU |
| PRR | Port Replacement Registers |
| PRU | Port Replacement Unit |
| EMULMEM | External emulation memory |
| access source | CPU or BDM or XGATE |

### 5.1.2 Features

The XEBI includes the following features:

- Output of up to 23-bit address bus and control signals to be used with a non-muxed external bus
- Bidirectional 16-bit external data bus with option to disable upper half
- Visibility of internal bus activity

### 5.1.3 Modes of Operation

- Single-chip modes

  The external bus interface is not available in these modes.
- Expanded modes

  Address, data, and control signals are activated on the external bus in normal expanded mode and special test mode.
- Emulation modes

  The external bus is activated to interface to an external tool for emulation of normal expanded mode or normal single-chip mode applications.

Refer to the S12X_MMC section for a detailed description of the MCU operating modes.

### 5.1.4 Block Diagram

Figure 5-1 is a block diagram of the XEBI with all related I/O signals.

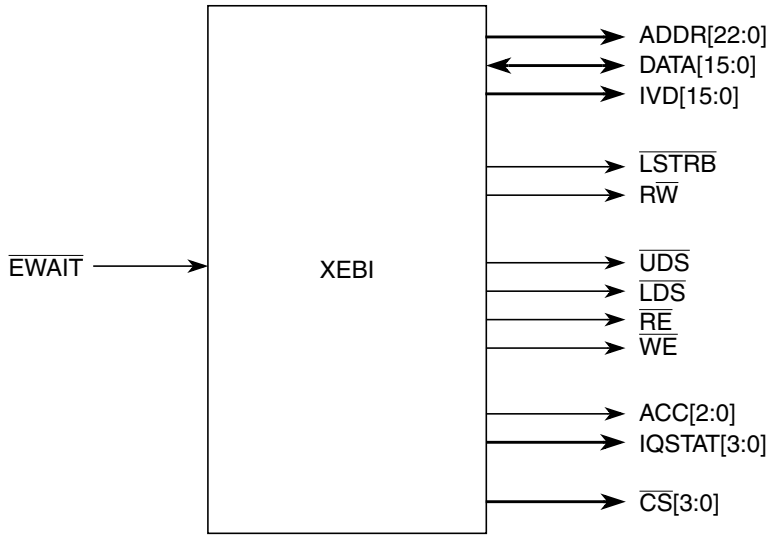



**Figure 5-1. XEBI Block Diagram**

## 5.2 External Signal Description

The user is advised to refer to the SoC section for port configuration and location of external bus signals.

**NOTE**

The following external bus related signals are described in other sections:
ECLK, ECLKX2 (free-running clocks) — PIM section
TAGHI, TAGLO (tag inputs) — PIM section, S12X_DBG section

Table 5-2 outlines the pin names and gives a brief description of their function. Refer to the SoC section and PIM section for reset states of these pins and associated pull-ups or pull-downs.

**Table 5-2. External System Signals Associated with XEBI**

| Signal | I[1]/O | EBI Signal Multiplex (T)ime[2] (F)unction[3] | | Description | Available in Modes | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | NS | SS | NX | ES | EX | ST |
| $\overline{RE}$ | O | — | — | Read Enable, indicates external read access | No | No | Yes | No | No | No |
| ADDR[22:20] | O | T | — | External address | No | No | Yes | Yes | Yes | Yes |
| ACC[2:0] | O | | — | Access source | No | No | No | Yes | Yes | Yes |
| ADDR[19:16] | O | T | — | External address | No | No | Yes | Yes | Yes | Yes |
| IQSTAT[3:0] | O | | — | Instruction Queue Status | No | No | No | Yes | Yes | Yes |
| ADDR[15:1] | O | T | — | External address | No | No | Yes | Yes | Yes | Yes |
| IVD[15:1] | O | | — | Internal visibility read data | No | No | No | Yes | Yes | Yes |
| ADDR0 | O | T | F | External address | No | No | No | Yes | Yes | Yes |
| IVD0 | O | | | Internal visibility read data | No | No | No | Yes | Yes | Yes |
| $\overline{UDS}$ | O | — | | Upper Data Select, indicates external access to the high byte DATA[15:8] | No | No | Yes | No | No | No |
| $\overline{LSTRB}$ | O | — | F | Low Strobe, indicates valid data on DATA[7:0] | No | No | No | Yes | Yes | Yes |
| $\overline{LDS}$ | O | — | | Lower Data Select, indicates external access to the low byte DATA[7:0] | No | No | Yes | No | No | No |
| $\overline{RW}$ | O | — | F | Read/Write, indicates the direction of internal data transfers | No | No | No | Yes | Yes | Yes |
| $\overline{WE}$ | O | — | | Write Enable, indicates external write access | No | No | Yes | No | No | No |
| $\overline{CS}$[3:0] | O | — | — | Chip select | No | No | Yes | No | Yes | No |
| DATA[15:8] | I/O | — | — | Bidirectional data (even address) | No | No | Yes | Yes | Yes | Yes |
| DATA[7:0] | I/O | — | — | Bidirectional data (odd address) | No | No | Yes | Yes | Yes | Yes |
| $\overline{EWAIT}$ | I | — | — | External control for external bus access stretches (adding wait states) | No | No | Yes | No | Yes | No |

1. All inputs are capable of reducing input threshold level
2. Time-multiplex means that the respective signals share the same pin on chip level and are active alternating in a dedicated time slot (in modes where applicable).
3. Function-multiplex means that one of the respective signals sharing the same pin on chip level continuously uses the pin depending on configuration and reset state.

## 5.3 Memory Map and Register Definition

This section provides a detailed description of all registers accessible in the XEBI.

### 5.3.1 Module Memory Map

The registers associated with the XEBI block are shown in Figure 5-2.

| Register Name | | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|---|
| 0x0E EBICTL0 | R | ITHRS | 0 | HDBE | ASIZ4 | ASIZ3 | ASIZ2 | ASIZ1 | ASIZ0 |
| | W | | | | | | | | |
| 0x0F EBICTL1 | R | 0 | EXSTR12 | EXSTR11 | EXSTR10 | 0 | EXSTR02 | EXSTR01 | EXSTR00 |
| | W | | | | | | | | |

= Unimplemented or Reserved

**Figure 5-2. XEBI Register Summary**

### 5.3.2 Register Descriptions

The following sub-sections provide a detailed description of each register and the individual register bits.

All control bits can be written anytime, but this may have no effect on the related function in certain operating modes. This allows specific configurations to be set up before changing into the target operating mode.

**NOTE**

Depending on the operating mode an available function may be enabled, disabled or depend on the control register bit. Reading the register bits will reflect the status of related function only if the current operating mode allows user control. Please refer the individual bit descriptions.

#### 5.3.2.1 External Bus Interface Control Register 0 (EBICTL0)

Module Base +0x000E (PRR)

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | ITHRS | 0 | HDBE | ASIZ4 | ASIZ3 | ASIZ2 | ASIZ1 | ASIZ0 |
| W | | | | | | | | |
| Reset | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 |

= Unimplemented or Reserved

**Figure 5-3. External Bus Interface Control Register 0 (EBICTL0)**

Read: Anytime. In emulation modes, read operations will return the data from the external bus, in all other modes, the data is read from this register.

Write: Anytime. In emulation modes, write operations will also be directed to the external bus.

This register controls input pin threshold level and determines the external address and data bus sizes in normal expanded mode. If not in use with the external bus interface, the related pins can be used for alternative functions.

External bus is available as programmed in normal expanded mode and always full-sized in emulation modes and special test mode; function not available in single-chip modes.

**Table 5-3. EBICTL0 Field Descriptions**

| Field | Description |
|---|---|
| 7<br>ITHRS | **Reduced Input Threshold** — This bit selects reduced input threshold on external data bus pins and specific control input signals which are in use with the external bus interface in order to adapt to external devices with a 3.3 V, 5 V tolerant I/O.<br>The reduced input threshold level takes effect depending on ITHRS, the operating mode and the related enable signals of the EBI pin function as summarized in Table 5-4.<br>0 Input threshold is at standard level on all pins<br>1 Reduced input threshold level enabled on pins in use with the external bus interface |
| 5<br>HDBE | **High Data Byte Enable** — This bit enables the higher half of the 16-bit data bus. If disabled, only the lower 8-bit data bus can be used with the external bus interface. In this case the unused data pins and the data select signals ($\overline{UDS}$ and $\overline{LDS}$) are free to be used for alternative functions.<br>0 DATA[15:8], $\overline{UDS}$, and $\overline{LDS}$ disabled<br>1 DATA[15:8], $\overline{UDS}$, and $\overline{LDS}$ enabled |
| 4–0<br>ASIZ[4:0] | **External Address Bus Size** — These bits allow scalability of the external address bus. The programmed value corresponds to the number of available low-aligned address lines (refer to Table 5-5). All address lines ADDR[22:0] start up as outputs after reset in expanded modes. This needs to be taken into consideration when using alternative functions on relevant pins in applications which utilize a reduced external address bus. |

**Table 5-4. Input Threshold Levels on External Signals**

| ITHRS | External Signal | NS | SS | NX | ES | EX | ST |
|---|---|---|---|---|---|---|---|
| 0 | DATA[15:8]<br>$\overline{TAGHI}$, $\overline{TAGLO}$ | Standard | Standard | Standard | Reduced | Reduced | Standard |
| | DATA[7:0] | | | | | | |
| | $\overline{EWAIT}$ | | | | Standard | Standard | |
| 1 | DATA[15:8]<br>$\overline{TAGHI}$, $\overline{TAGLO}$ | Standard | Standard | Reduced<br>if HDBE = 1 | Reduced | Reduced | Reduced |
| | DATA[7:0] | | | Reduced | | | |
| | $\overline{EWAIT}$ | | | Reduced<br>if EWAIT<br>enabled(1) | Standard | Reduced<br>if EWAIT<br>enabled[1] | Standard |

1. EWAIT function is enabled if at least one $\overline{CSx}$ line is configured respectively in MMCCTL0. Refer to S12X_MMC section and Table 5-6.

**Table 5-5. External Address Bus Size**

| ASIZ[4:0] | Available External Address Lines |
|---|---|
| 00000 | None |
| 00001 | $\overline{UDS}$ |
| 00010 | ADDR1, $\overline{UDS}$ |

**Table 5-5. External Address Bus Size**

| ASIZ[4:0] | Available External Address Lines |
|---|---|
| 00011 | ADDR[2:1], $\overline{\text{UDS}}$ |
| : | : |
| 10110 | ADDR[21:1], $\overline{\text{UDS}}$ |
| 10111<br>:<br>11111 | ADDR[22:1], $\overline{\text{UDS}}$ |

### 5.3.2.2 External Bus Interface Control Register 1 (EBICTL1)

Module Base +0x000F (PRR)

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | 0 | EXSTR12 | EXSTR11 | EXSTR10 | 0 | EXSTR02 | EXSTR01 | EXSTR00 |
| W | | | | | | | | |
| Reset | 0 | 1 | 1 | 1 | 0 | 1 | 1 | 1 |

= Unimplemented or Reserved

**Figure 5-4. External Bus Interface Control Register 1 (EBICTL1)**

Read: Anytime. In emulation modes, read operations will return the data from the external bus, in all other modes the data is read from this register.

Write: Anytime. In emulation modes, write operations will also be directed to the external bus.

This register allows programming of two independent values determining the amount of additional stretch cycles for external accesses (wait states).

With two bits in S12X_MMC register MMCCTL0 for every individual $\overline{\text{CSx}}$ line one of the two counter options or the $\overline{\text{EWAIT}}$ input is selected as stretch source. The chip select outputs can also be disabled to free up the pins for alternative functions (Table 5-6). Refer also to S12X_MMC section for register bit descriptions.

**Table 5-6. Chip select function**

| CSxE1 | CSxE0 | Function |
|---|---|---|
| 0 | 0 | $\overline{\text{CSx}}$ disabled |
| 0 | 1 | $\overline{\text{CSx}}$ stretched with EXSTR0 |
| 1 | 0 | $\overline{\text{CSx}}$ stretched with EXSTR1 |
| 1 | 1 | $\overline{\text{CSx}}$ stretched with $\overline{\text{EWAIT}}$ |

If $\overline{\text{EWAIT}}$ input usage is selected in MMCCTL0 the minimum number of stretch cycles is 2 for accesses to the related address range.

If configured respectively, stretch cycles are added as programmed or dependent on $\overline{\text{EWAIT}}$ in normal expanded mode and emulation expanded mode; function not available in all other operating modes.

**Table 5-7. EBICTL1 Field Descriptions**

| Field | Description |
|---|---|
| 6–4<br>EXSTR1[2:0] | **External Access Stretch Option 1 Bits 2, 1, 0** — This three bit field determines the amount of additional clock stretch cycles on every access to the external address space as shown in Table 5-8. |
| 2–0<br>EXSTR0[2:0] | **External Access Stretch Option 0 Bits 2, 1, 0** — This three bit field determines the amount of additional clock stretch cycles on every access to the external address space as shown in Table 5-8. |

**Table 5-8. External Access Stretch Bit Definition**

| EXSTRx[2:0] | Number of Stretch Cycles |
|---|---|
| 000 | 1 |
| 001 | 2 |
| 010 | 3 |
| 011 | 4 |
| 100 | 5 |
| 101 | 6 |
| 110 | 7 |
| 111 | 8 |

## 5.4 Functional Description

This section describes the functions of the external bus interface. The availability of external signals and functions in relation to the operating mode is initially summarized and described in more detail in separate sub-sections.

### 5.4.1 Operating Modes and External Bus Properties

A summary of the external bus interface functions for each operating mode is shown in Table 5-9.

**Table 5-9. Summary of Functions**

| Properties (if Enabled) | Single-Chip Modes | | Expanded Modes | | | |
|---|---|---|---|---|---|---|
| | Normal Single-Chip | Special Single-Chip | Normal Expanded | Emulation Single-Chip | Emulation Expanded | Special Test |
| **Timing Properties** | | | | | | |
| PRR access[1] | 2 cycles<br>read internal<br>write internal | 2 cycles<br>read internal<br>write internal | 2 cycles<br>read internal<br>write internal | 2 cycles<br>read external<br>write int & ext | 2 cycles<br>read external<br>write int & ext | 2 cycles<br>read internal<br>write internal |
| Internal access visible externally | — | — | — | 1 cycle | 1 cycle | 1 cycle |
| External address access and unimplemented area access[2] | — | — | Max. of 2 to 9 programmed cycles<br>or n cycles of ext. wait[3] | 1 cycle | Max. of 2 to 9 programmed cycles<br>or n cycles of ext. wait[3] | 1 cycle |

**Table 5-9. Summary of Functions (continued)**

| Properties (if Enabled) | Single-Chip Modes | | Expanded Modes | | | |
|---|---|---|---|---|---|---|
| | Normal Single-Chip | Special Single-Chip | Normal Expanded | Emulation Single-Chip | Emulation Expanded | Special Test |
| Flash area address access[4] | — | — | — | 1 cycle | 1 cycle | 1 cycle |
| **Signal Properties** | | | | | | |
| Bus signals | — | — | ADDR[22:1]<br>DATA[15:0] | ADDR[22:20]/<br>ACC[2:0]<br>ADDR[19:16]/<br>IQSTAT[3:0]<br>ADDR[15:0]/<br>IVD[15:0]<br>DATA[15:0] | ADDR[22:20]/<br>ACC[2:0]<br>ADDR[19:16]/<br>IQSTAT[3:0]<br>ADDR[15:0]/<br>IVD[15:0]<br>DATA[15:0] | ADDR[22:0]<br>DATA[15:0] |
| Data select signals (if 16-bit data bus) | — | — | $\overline{UDS}$<br>$\overline{LDS}$ | ADDR0<br>$\overline{LSTRB}$ | ADDR0<br>$\overline{LSTRB}$ | ADDR0<br>$\overline{LSTRB}$ |
| Data direction signals | — | — | $\overline{RE}$<br>$\overline{WE}$ | R$\overline{W}$ | R$\overline{W}$ | R$\overline{W}$ |
| Chip Selects | — | — | $\overline{CS0}$<br>$\overline{CS1}$<br>$\overline{CS2}$<br>$\overline{CS3}$ | — | $\overline{CS0}$<br>$\overline{CS1}$<br>$\overline{CS2}$<br>$\overline{CS3}$ | — |
| External wait feature | — | — | $\overline{EWAIT}$ | — | $\overline{EWAIT}$ | — |
| Reduced input threshold enabled on | — | — | Refer to Table 5-4 | DATA[15:0]<br>$\overline{EWAIT}$ | DATA[15:0]<br>$\overline{EWAIT}$ | Refer to Table 5-4 |

1. Incl. S12X_EBI registers
2. Refer to S12X_MMC section.
3. If EWAIT enabled for at least one $\overline{CSx}$ line (refer to S12X_MMC section), the minimum number of external bus cycles is 3.
4. Available only if configured appropriately by ROMON and EROMON (refer to S12X_MMC section).

## 5.4.2 Internal Visibility

Internal visibility allows the observation of the internal CPU address and data bus as well as the determination of the access source and the CPU pipe (queue) status through the external bus interface.

Internal visibility is always enabled in emulation single chip mode and emulation expanded mode. Internal CPU accesses are made visible on the external bus interface except CPU execution of BDM firmware instructions.

Internal reads are made visible on ADDRx/IVDx (address and read data multiplexed, see Table 5-12 to Table 5-14), internal writes on ADDRx and DATAx (see Table 5-15 to Table 5-17). R$\overline{W}$ and $\overline{LSTRB}$ show the type of access. External read data are also visible on IVDx.

During ‘no access’ cycles R$\overline{W}$ is held in read position while $\overline{LSTRB}$ is undetermined.

All accesses which make use of the external bus interface are considered external accesses.

### 5.4.2.1 Access Source Signals (ACC)

The access source can be determined from the external bus control signals ACC[2:0] as shown in Table 5-10.

**Table 5-10. Determining Access Source from Control Signals**

| ACC[2:0] | Access Description |
|---|---|
| 000 | Repetition of previous access cycle |
| 001 | CPU access |
| 010 | BDM external access |
| 011 | XGATE PRR access |
| 100 | No access(1) |
| 101 | CPU access error |
| 110, 111 | Reserved |

1. Denotes also CPU accesses to BDM firmware and BDM registers (IQSTATx are 'XXXX' and $\overline{RW}$ = 1 in these cases)

### 5.4.2.2 Instruction Queue Status Signals (IQSTAT)

The CPU instruction queue status (execution-start and data-movement information) is brought out as IQSTAT[3:0] signals. For decoding of the IQSTAT values, refer to the S12X_CPU section.

### 5.4.2.3 Internal Visibility Data (IVD)

Depending on the access size and alignment, either a word of read data is made visible on the address lines or only the related data byte will be presented in the ECLK low phase. For details refer to Table 5-11.

Invalid IVD are brought out in case of non-CPU read accesses.

**Table 5-11. IVD Read Data Output**

| Access | IVD[15:8] | IVD[7:0] |
|---|---|---|
| Word read of data at an even and even+1 address | ivd(even) | ivd(even+1) |
| Word read of data at an odd and odd+1 internal RAM address (misaligned) | ivd(odd+1) | ivd(odd) |
| Byte read of data at an even address | ivd(even) | addr[7:0] (rep.) |
| Byte read of data at an odd address | addr[15:8] (rep.) | ivd(odd) |

### 5.4.2.4 Emulation Modes Timing

A bus access lasts 1 ECLK cycle. In case of a stretched external access (emulation expanded mode), up to an infinite amount of ECLK cycles may be added. ADDRx values will only be shown in ECLK high phases, while ACCx, IQSTATx, and IVDx values will only be presented in ECLK low phases.

Based on this multiplex timing, ACCx are only shown in the current (first) access cycle. IQSTATx and (for read accesses) IVDx follow in the next cycle. If the access takes more than one bus cycle, ACCx display NULL (0x000) in the second and all following cycles of the access. IQSTATx display NULL (0x0000) from the third until one cycle after the access to indicate continuation.

The resulting timing pattern of the external bus signals is outlined in the following tables for read, write and interleaved read/write accesses. Three examples represent different access lengths of 1, 2, and n–1 bus cycles. Non-shaded bold entries denote all values related to Access #0.

The following terminology is used:

‘addr’ — value(ADDRx); small letters denote the logic values at the respective pins
‘x’ — Undefined output pin values
‘z’ — Tristate pins
‘?’ — Dependent on previous access (read or write); IVDx: ‘ivd’ or ‘x’; DATAx: ‘data’ or ‘z’

#### 5.4.2.4.1 Read Access Timing

**Table 5-12. Read Access (1 Cycle)**

| | | Access #0 | | | | Access #1 | | |
|---|---|---|---|---|---|---|---|---|
| **Bus cycle ->** | ... | **1** | | **2** | | **3** | | ... |
| ECLK phase | ... | high | low | high | low | high | low | ... |
| ADDR[22:20] / ACC[2:0] | ... | | **acc 0** | | acc 1 | | acc 2 | ... |
| ADDR[19:16] / IQSTAT[3:0] | ... | **addr 0** | iqstat -1 | addr 1 | **iqstat 0** | addr 2 | iqstat 1 | ... |
| ADDR[15:0] / IVD[15:0] | ... | | ? | | **ivd 0** | | ivd 1 | ... |
| DATA[15:0] (internal read) | ... | ? | **z** | **z** | z | z | z | ... |
| DATA[15:0] (external read) | ... | ? | **z** | **data 0** | z | data 1 | z | ... |
| R$\overline{W}$ | ... | 1 | 1 | 1 | 1 | 1 | 1 | ... |

**Table 5-13. Read Access (2 Cycles)**

| | | Access #0 | | | | Access #1 | | |
|---|---|---|---|---|---|---|---|---|
| **Bus cycle ->** | ... | **1** | | **2** | | **3** | | ... |
| ECLK phase | ... | high | low | high | low | high | low | ... |
| ADDR[22:20] / ACC[2:0] | ... | | **acc 0** | | **000** | | acc 1 | ... |
| ADDR[19:16] / IQSTAT[3:0] | ... | **addr 0** | iqstat-1 | **addr 0** | **iqstat 0** | addr 1 | **0000** | ... |
| ADDR[15:0] / IVD[15:0] | ... | | ? | | **x** | | **ivd 0** | ... |
| DATA[15:0] (internal read) | ... | ? | **z** | **z** | **z** | **z** | z | ... |
| DATA[15:0] (external read) | ... | ? | **z** | **z** | **z** | **data 0** | z | ... |
| R$\overline{W}$ | ... | 1 | 1 | 1 | 1 | 1 | 1 | ... |

**Table 5-14. Read Access (n–1 Cycles)**

| | | Access #0 | | | | | | | Access #1 | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| **Bus cycle ->** | ... | **1** | | **2** | | **3** | | ... | **n** | | ... |
| ECLK phase | ... | high | low | high | low | high | low | ... | high | low | ... |
| ADDR[22:20] / ACC[2:0] | ... | | **acc 0** | | **000** | | **000** | ... | | acc 1 | ... |
| ADDR[19:16] / IQSTAT[3:0] | ... | **addr 0** | iqstat-1 | **addr 0** | **iqstat 0** | **addr 0** | **0000** | ... | addr 1 | **0000** | ... |
| ADDR[15:0] / IVD[15:0] | ... | | ? | | **x** | | **x** | ... | | **ivd 0** | ... |
| DATA[15:0] (internal read) | ... | ? | **z** | **z** | **z** | **z** | **z** | ... | **z** | z | ... |

**Table 5-14. Read Access (n–1 Cycles)**

| DATA[15:0] (external read) | ... | ? | **z** | **z** | **z** | **z** | **z** | ... | **data 0** | z | ... |
|---|---|---|---|---|---|---|---|---|---|---|---|
| R$\overline{W}$ | ... | 1 | 1 | 1 | 1 | 1 | 1 | ... | 1 | 1 | ... |

### 5.4.2.4.2 Write Access Timing

**Table 5-15. Write Access (1 Cycle)**

| | | **Access #0** | | **Access #1** | | **Access #2** | | |
|---|---|---|---|---|---|---|---|---|
| **Bus cycle ->** | ... | **1** | | **2** | | **3** | | ... |
| ECLK phase | ... | high | low | high | low | high | low | ... |
| ADDR[22:20] / ACC[2:0] | ... | | **acc 0** | | acc 1 | | acc 2 | ... |
| ADDR[19:16] / IQSTAT[3:0] | ... | **addr 0** | iqstat -1 | addr 1 | **iqstat 0** | addr 2 | iqstat 1 | ... |
| ADDR[15:0] / IVD[15:0] | ... | | ? | | **x** | | x | ... |
| DATA[15:0] (write) | ... | ? | **data 0** | | data 1 | | data 2 | ... |
| R$\overline{W}$ | ... | 0 | 0 | 1 | 1 | 1 | 1 | ... |

**Table 5-16. Write Access (2 Cycles)**

| | | **Access #0** | | | | **Access #1** | | |
|---|---|---|---|---|---|---|---|---|
| **Bus cycle ->** | ... | **1** | | **2** | | **3** | | ... |
| ECLK phase | ... | high | low | high | low | high | low | ... |
| ADDR[22:20] / ACC[2:0] | ... | | **acc 0** | | **000** | | acc 1 | ... |
| ADDR[19:16] / IQSTAT[3:0] | ... | **addr 0** | iqstat-1 | **addr 0** | **iqstat 0** | addr 1 | **0000** | ... |
| ADDR[15:0] / IVD[15:0] | ... | | ? | | **x** | | **x** | ... |
| DATA[15:0] (write) | ... | ? | **data 0** | | | | x | ... |
| R$\overline{W}$ | ... | 0 | 0 | 0 | 0 | 1 | 1 | ... |

**Table 5-17. Write Access (n–1 Cycles)**

| | | **Access #0** | | | | | | | **Access #1** | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| **Bus cycle ->** | ... | **1** | | **2** | | **3** | | ... | **n** | | ... |
| ECLK phase | ... | high | low | high | low | high | low | ... | high | low | ... |
| ADDR[22:20] / ACC[2:0] | ... | | **acc 0** | | **000** | | **000** | **...** | | acc 1 | ... |
| ADDR[19:16] / IQSTAT[3:0] | ... | **addr 0** | iqstat-1 | **addr 0** | **iqstat 0** | **addr 0** | **0000** | **...** | addr 1 | **0000** | ... |
| ADDR[15:0] / IVD[15:0] | ... | | ? | | **x** | | **x** | **...** | | **x** | ... |
| DATA[15:0] (write) | ... | ? | **data 0** | | | | | | | x | ... |
| R$\overline{W}$ | ... | 0 | 0 | 0 | 0 | 0 | 0 | ... | 1 | 1 | ... |

### 5.4.2.4.3 Read-Write-Read Access Timing

**Table 5-18. Interleaved Read-Write-Read Accesses (1 Cycle)**

| | | **Access #0** | **Access #1** | **Access #2** | |
|---|---|---|---|---|---|
| **Bus cycle ->** | ... | **1** | **2** | **3** | ... |

**Table 5-18. Interleaved Read-Write-Read Accesses (1 Cycle) (continued)**

| ECLK phase | ... | high | low | high | low | high | low | ... |
|---|---|---|---|---|---|---|---|---|
| ADDR[22:20] / ACC[2:0] | ... | **addr 0** | **acc 0** | addr 1 | acc 1 | addr 2 | acc 2 | ... |
| ADDR[19:16] / IQSTAT[3:0] | ... | | iqstat -1 | | **iqstat 0** | | iqstat 1 | ... |
| ADDR[15:0] / IVD[15:0] | ... | | ? | | **ivd 0** | | x | ... |
| DATA[15:0] (internal read) | ... | ? | **z** | **z** | (write) data 1 | | z | ... |
| DATA[15:0] (external read) | ... | ? | **z** | **data 0** | (write) data 1 | | z | ... |
| R$\overline{W}$ | ... | 1 | 1 | 0 | 0 | 1 | 1 | ... |

## 5.4.3 Accesses to Port Replacement Registers

All read and write accesses to PRR addresses take two bus clock cycles independent of the operating mode. If writing to these addresses in emulation modes, the access is directed to both, the internal register and the external resource while reads will be treated external.

The XEBI control registers also belong to this category.

## 5.4.4 Stretched External Bus Accesses

In order to allow fast internal bus cycles to coexist in a system with slower external resources, the XEBI supports stretched external bus accesses (wait states) for each external address range related to one of the 4 chip select lines individually.

This feature is available in normal expanded mode and emulation expanded mode for accesses to all external addresses except emulation memory and PRR. In these cases the fixed access times are 1 or 2 cycles, respectively.

Stretched accesses are controlled by:

1. EXSTR1[2:0] and EXSTR0[2:0] bits in the EBICTL1 register configuring a fixed amount of stretch cycles individually for each $\overline{CSx}$ line in MMCCTL0
2. Activation of the external wait feature for each $\overline{CSx}$ line MMCCTL0 register
3. Assertion of the external $\overline{EWAIT}$ signal when at least one $\overline{CSx}$ line is configured for EWAIT

The EXSTRx[2:0] control bits can be programmed for generation of a fixed number of 1 to 8 stretch cycles. If the external wait feature is enabled, the minimum number of additional stretch cycles is 2. An arbitrary amount of stretch cycles can be added using the $\overline{EWAIT}$ input.

$\overline{EWAIT}$ needs to be asserted at least for a minimal specified time window within an external access cycle for the internal logic to detect it and add a cycle (refer to electrical characteristics). Holding it for additional cycles will cause the external bus access to be stretched accordingly.

Write accesses are stretched by holding the initiator in its current state for additional cycles as programmed and controlled by external wait after the data have been driven out on the external bus. This results in an extension of time the bus signals and the related control signals are valid externally.

Read data are not captured by the system in normal expanded mode until the specified setup time before the $\overline{RE}$ rising edge.

Read data are not captured in emulation expanded mode until the specified setup time before the falling edge of ECLK.

In emulation expanded mode, accesses to the internal flash or the emulation memory (determined by EROMON and ROMON bits; see S12X_MMC section for details) always take 1 cycle and stretching is not supported. In case the internal flash is taken out of the map in user applications, accesses are stretched as programmed and controlled by external wait.

## 5.4.5 Data Select and Data Direction Signals

The S12X_EBI supports byte and word accesses at any valid external address. The big endian system of the MCU is extended to the external bus; however, word accesses are restricted to even aligned addresses. The only exception is the visibility of misaligned word accesses to addresses in the internal RAM as this module exclusively supports these kind of accesses in a single cycle.

With the above restriction, a fixed relationship is implied between the address parity and the dedicated bus halves where the data are accessed: DATA[15:8] is related to even addresses and DATA[7:0] is related to odd addresses.

In expanded modes the data access type is externally determined by a set of control signals, i.e., data select and data direction signals, as described below. The data select signals are not available if using the external bus interface with an 8-bit data bus.

### 5.4.5.1 Normal Expanded Mode

In normal expanded mode, the external signals $\overline{RE}$, $\overline{WE}$, $\overline{UDS}$, $\overline{LDS}$ indicate the access type (read/write), data size and alignment of an external bus access (Table 5-19).

**Table 5-19. Access in Normal Expanded Mode**

| Access | $\overline{RE}$ | $\overline{WE}$ | $\overline{UDS}$ | $\overline{LDS}$ | DATA[15:8] | | DATA[7:0] | |
|---|---|---|---|---|---|---|---|---|
| | | | | | I/O | data(addr) | I/O | data(addr) |
| Word write of data on DATA[15:0] at an even and even+1 address | 1 | 0 | 0 | 0 | Out | data(even) | Out | data(odd) |
| Byte write of data on DATA[7:0] at an odd address | 1 | 0 | 1 | 0 | In | x | Out | data(odd) |
| Byte write of data on DATA[15:8] at an even address | 1 | 0 | 0 | 1 | Out | data(even) | In | x |
| Word read of data on DATA[15:0] at an even and even+1 address | 0 | 1 | 0 | 0 | In | data(even) | In | data(odd) |
| Byte read of data on DATA[7:0] at an odd address | 0 | 1 | 1 | 0 | In | x | In | data(odd) |
| Byte read of data on DATA[15:8] at an even address | 0 | 1 | 0 | 1 | In | data(even) | In | x |
| Indicates No Access | 1 | 1 | 1 | 1 | In | x | In | x |
| Unimplemented | 1 | 1 | 1 | 0 | In | x | In | x |
| | 1 | 1 | 0 | 1 | In | x | In | x |

### 5.4.5.2 Emulation Modes and Special Test Mode

In emulation modes and special test mode, the external signals $\overline{LSTRB}$, $R\overline{W}$, and ADDR0 indicate the access type (read/write), data size and alignment of an external bus access. Misaligned accesses to the

internal RAM and misaligned XGATE PRR accesses in emulation modes are the only type of access that are able to produce $\overline{LSTRB}$ = ADDR0 = 1. This is summarized in Table 5-20.

**Table 5-20. Access in Emulation Modes and Special Test Mode**

| Access | R/$\overline{W}$ | $\overline{LSTRB}$ | ADDR0 | DATA[15:8] | | DATA[7:0] | |
|---|---|---|---|---|---|---|---|
| | | | | I/O | data(addr) | I/O | data(addr) |
| Word write of data on DATA[15:0] at an even and even+1 address | 0 | 0 | 0 | Out | data(even) | Out | data(odd) |
| Byte write of data on DATA[7:0] at an odd address | 0 | 0 | 1 | In | x | Out | data(odd) |
| Byte write of data on DATA[15:8] at an even address | 0 | 1 | 0 | Out | data(odd) | In | x |
| Word write at an odd and odd+1 internal RAM address (misaligned — only in emulation modes) | 0 | 1 | 1 | Out | data(odd+1) | Out | data(odd) |
| Word read of data on DATA[15:0] at an even and even+1 address | 1 | 0 | 0 | In | data(even) | In | data(even+1) |
| Byte read of data on DATA[7:0] at an odd address | 1 | 0 | 1 | In | x | In | data(odd) |
| Byte read of data on DATA[15:8] at an even address | 1 | 1 | 0 | In | data(even) | In | x |
| Word read at an odd and odd+1 internal RAM address (misaligned - only in emulation modes) | 1 | 1 | 1 | In | data(odd+1) | In | data(odd) |

### 5.4.6 Low-Power Options

The XEBI does not support any user-controlled options for reducing power consumption.

#### 5.4.6.1 Run Mode

The XEBI does not support any options for reducing power in run mode.

Power consumption is reduced in single-chip modes due to the absence of the external bus interface. Operation in expanded modes results in a higher power consumption, however any unnecessary toggling of external bus signals is reduced to the lowest indispensable activity by holding the previous states between external accesses.

#### 5.4.6.2 Wait Mode

The XEBI does not support any options for reducing power in wait mode.

#### 5.4.6.3 Stop Mode

The XEBI will cease to function in stop mode.

## 5.5 Initialization/Application Information

This section describes the external bus interface usage and timing. Typical customer operating modes are normal expanded mode and emulation modes, specifically to be used in emulator applications. Taking the availability of the external wait feature into account the use cases are divided into four scenarios:

- Normal expanded mode
  - — External wait feature disabled
  - – External wait feature enabled
- Emulation modes
  - – Emulation single-chip mode (without wait states)
  - – Emulation expanded mode (with optional access stretching)

Normal single-chip mode and special single-chip mode do not have an external bus. Special test mode is used for factory test only. Therefore, these modes are omitted here.

All timing diagrams referred to throughout this section are available in the Electrical Characteristics appendix of the SoC section.

### 5.5.1 Normal Expanded Mode

This mode allows interfacing to external memories or peripherals which are available in the commercial market. In these applications the normal bus operation requires a minimum of 1 cycle stretch for each external access.

#### 5.5.1.1 Example 1a: External Wait Feature Disabled

The first example of bus timing of an external read and write access with the external wait feature disabled is shown in

- Figure ‘Example 1a: Normal Expanded Mode — Read Followed by Write’

The associated supply voltage dependent timing are numbers given in

- Table ‘Example 1a: Normal Expanded Mode Timing $V_{DD5}$ = 5.0 V (EWAIT disabled)’
- Table ‘Example 1a: Normal Expanded Mode Timing $V_{DD5}$ = 3.0 V (EWAIT disabled)’

Systems designed this way rely on the internal programmable access stretching. These systems have predictable external memory access times. The additional stretch time can be programmed up to 8 cycles to provide longer access times.

#### 5.5.1.2 Example 1b: External Wait Feature Enabled

The external wait operation is shown in this example. It can be used to exceed the amount of stretch cycles over the programmed number in EXSTR[2:0]. The feature must be enabled by configuring at least one $\overline{CSx}$ line for EWAIT.

If the $\overline{EWAIT}$ signal is not asserted, the number of stretch cycles is forced to a minimum of 2 cycles. If $\overline{EWAIT}$ is asserted within the predefined time window during the access it will be strobed active and another stretch cycle is added. If strobed inactive, the next cycle will be the last cycle before the access is finished. $\overline{EWAIT}$ can be held asserted as long as desired to stretch the access.

An access with 1 cycle stretch by $\overline{EWAIT}$ assertion is shown in

- Figure ‘Example 1b: Normal Expanded Mode — Stretched Read Access’
- Figure ‘Example 1b: Normal Expanded Mode — Stretched Write Access’

The associated timing numbers for both operations are given in

- Table ‘Example 1b: Normal Expanded Mode Timing $V_{DD5}$ = 5.0 V (EWAIT enabled)’
- Table ‘Example 1b: Normal Expanded Mode Timing $V_{DD5}$ = 3.0 V (EWAIT enabled)’

It is recommended to use the free-running clock (ECLK) at the fastest rate (bus clock rate) to synchronize the $\overline{EWAIT}$ input signal.

### 5.5.2 Emulation Modes

In emulation mode applications, the development systems use a custom PRU device to rebuild the single-chip or expanded bus functions which are lost due to the use of the external bus with an emulator.

Accesses to a set of registers controlling the related ports in normal modes (refer to SoC section) are directed to the external bus in emulation modes which are substituted by PRR as part of the PRU. Accesses to these registers take a constant time of 2 cycles.

Depending on the setting of ROMON and EROMON (refer to S12X_MMC section), the program code can be executed from internal memory or an optional external emulation memory (EMULMEM). No wait

state operation (stretching) of the external bus access is done in emulation modes when accessing internal memory or emulation memory addresses.

In both modes observation of the internal operation is supported through the external bus (internal visibility).

### 5.5.2.1 Example 2a: Emulation Single-Chip Mode

This mode is used for emulation systems in which the target application is operating in normal single-chip mode.

Figure 5-5 shows the PRU connection with the available external bus signals in an emulator application.

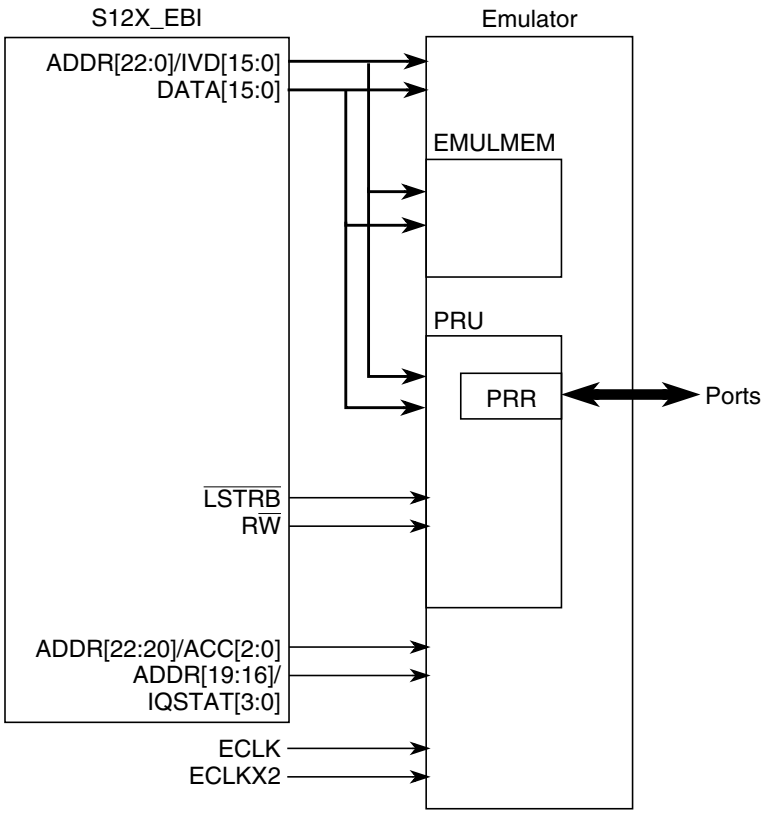



**Figure 5-5. Application in Emulation Single-Chip Mode**

The timing diagram for this operation is shown in:

- Figure ‘Example 2a: Emulation Single-Chip Mode — Read Followed by Write’

The associated timing numbers are given in:

- Table ‘Example 2a: Emulation Single-Chip Mode Timing (EWAIT disabled)’

Timing considerations:

- Signals muxed with address lines ADDRx, i.e., IVDx, IQSTATx and ACCx, have the same timing.
- $\overline{\text{LSTRB}}$ has the same timing as R$\overline{\text{W}}$.

- ECLKX2 rising edges have the same timing as ECLK edges.
- The timing for accesses to PRU registers, which take 2 cycles to complete, is the same as the timing for an external non-PRR access with 1 cycle of stretch as shown in example 2b.

### 5.5.2.2 Example 2b: Emulation Expanded Mode

This mode is used for emulation systems in which the target application is operating in normal expanded mode.

If the external bus is used with a PRU, the external device rebuilds the data select and data direction signals $\overline{UDS}$, $\overline{LDS}$, $\overline{RE}$, and $\overline{WE}$ from the ADDR0, $\overline{LSTRB}$, and R$\overline{W}$ signals.

Figure 5-6 shows the PRU connection with the available external bus signals in an emulator application.

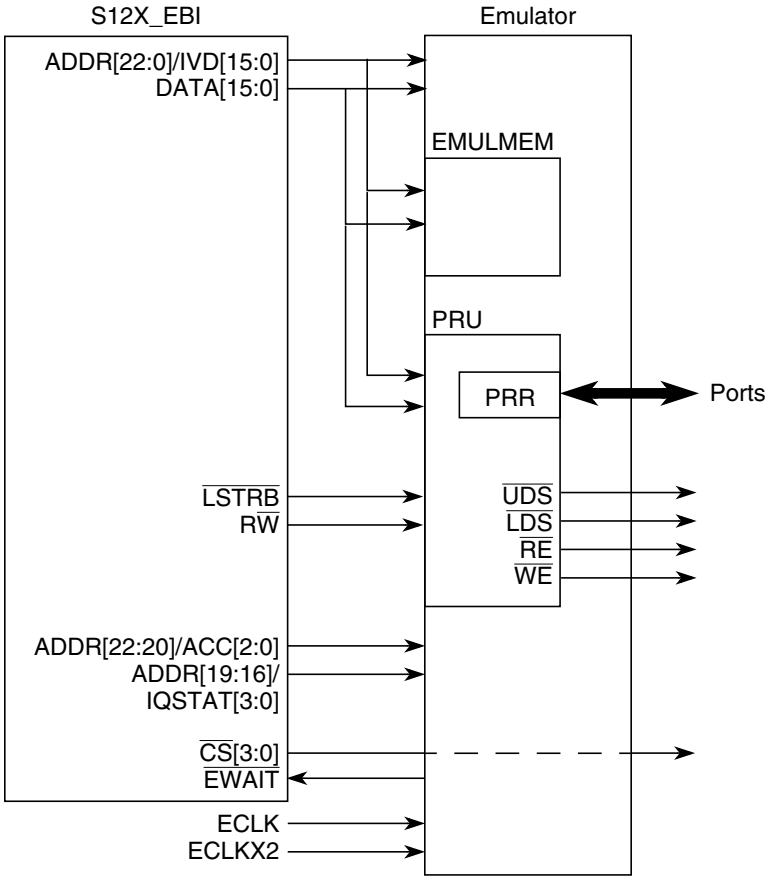



**Figure 5-6. Application in Emulation Expanded Mode**

The timings of accesses with 1 stretch cycle are shown in

- Figure ‘Example 2b: Emulation Expanded Mode — Read with 1 Stretch Cycle’
- Figure ‘Example 2b: Emulation Expanded Mode — Write with 1 Stretch Cycle’

The associated timing numbers are given in

- Table ‘Example 2b: Emulation Expanded Mode Timing $V_{DD5}$ = 5.0 V (EWAIT disabled)’ (this also includes examples for alternative settings of 2 and 3 additional stretch cycles)

Timing considerations:

- If no stretch cycle is added, the timing is the same as in Emulation Single-Chip Mode.

# Chapter 6
# Interrupt (S12XINTV2)

**Table 6-1. Revision History**

| Revision Number | Revision Date | Sections Affected | Description of Changes |
|---|---|---|---|
| V02.00 | 01 Jul 2005 | 6.1.2/6-262 | Initial V2 release, added new features:<br>- XGATE threads can be interrupted.<br>- SYS instruction vector.<br>- Access violation interrupt vectors. |
| V02.04 | 11 Jan 2007 | 6.3.2.2/6-267<br>6.3.2.4/6-268 | - Added Notes for devices without XGATE module. |
| V02.05 | 20 Mar 2007 | 6.4.6/6-274 | - Fixed priority definition for software exceptions. |
| V02.07 | 13 Dec 2011 | 6.5.3.1/6-276 | - Re-worded for difference of Wake-up feature between STOP and WAIT modes. |

## 6.1 Introduction

The XINT module decodes the priority of all system exception requests and provides the applicable vector for processing the exception to either the CPU or the XGATE module. The XINT module supports:

- I bit and X bit maskable interrupt requests
- One non-maskable unimplemented op-code trap
- One non-maskable software interrupt (SWI) or background debug mode request
- One non-maskable system call interrupt (SYS)
- Three non-maskable access violation interrupts
- One spurious interrupt vector request
- Three system reset vector requests

Each of the I bit maskable interrupt requests can be assigned to one of seven priority levels supporting a flexible priority scheme. For interrupt requests that are configured to be handled by the CPU, the priority scheme can be used to implement nested interrupt capability where interrupts from a lower level are automatically blocked if a higher level interrupt is being processed. Interrupt requests configured to be handled by the XGATE module can be nested one level deep.

**NOTE**

The HPRIO register and functionality of the original S12 interrupt module is no longer supported. It is superseded by the 7-level interrupt request priority scheme.

### 6.1.1 Glossary

The following terms and abbreviations are used in the document.

**Table 6-2. Terminology**

| Term | Meaning |
|---|---|
| CCR | Condition Code Register (in the S12X CPU) |
| DMA | Direct Memory Access |
| INT | Interrupt |
| IPL | Interrupt Processing Level |
| ISR | Interrupt Service Routine |
| MCU | Micro-Controller Unit |
| XGATE | refers to the XGATE co-processor; XGATE is an optional feature |
| $\overline{\text{IRQ}}$ | refers to the interrupt request associated with the $\overline{\text{IRQ}}$ pin |
| $\overline{\text{XIRQ}}$ | refers to the interrupt request associated with the $\overline{\text{XIRQ}}$ pin |

### 6.1.2 Features

- Interrupt vector base register (IVBR)
- One spurious interrupt vector (at address vector base[1] + 0x0010).
- One non-maskable system call interrupt vector request (at address vector base + 0x0012).
- Three non-maskable access violation interrupt vector requests (at address vector base + 0x0014–0x0018).
- 2–109 I bit maskable interrupt vector requests (at addresses vector base + 0x001A–0x00F2).
- Each I bit maskable interrupt request has a configurable priority level and can be configured to be handled by either the CPU or the XGATE module[2].
- I bit maskable interrupts can be nested, depending on their priority levels.
- One X bit maskable interrupt vector request (at address vector base + 0x00F4).
- One non-maskable software interrupt request (SWI) or background debug mode vector request (at address vector base + 0x00F6).
- One non-maskable unimplemented op-code trap (TRAP) vector (at address vector base + 0x00F8).
- Three system reset vectors (at addresses 0xFFFA–0xFFFE).
- Determines the highest priority XGATE and interrupt vector requests, drives the vector to the XGATE module or to the bus on CPU request, respectively.
- Wakes up the system from stop or wait mode when an appropriate interrupt request occurs or whenever $\overline{\text{XIRQ}}$ is asserted, even if X interrupt is masked.
- XGATE can wake up and execute code, even with the CPU remaining in stop or wait mode.

1. The vector base is a 16-bit address which is accumulated from the contents of the interrupt vector base register (IVBR, used as upper byte) and 0x00 (used as lower byte).
2. The $\overline{\text{IRQ}}$ interrupt can only be handled by the CPU

### 6.1.3 Modes of Operation

- Run mode

  This is the basic mode of operation.
- Wait mode

  In wait mode, the XINT module is frozen. It is however capable of either waking up the CPU if an interrupt occurs or waking up the XGATE if an XGATE request occurs. Please refer to Section 6.5.3, “Wake Up from Stop or Wait Mode” for details.
- Stop Mode

  In stop mode, the XINT module is frozen. It is however capable of either waking up the CPU if an interrupt occurs or waking up the XGATE if an XGATE request occurs. Please refer to Section 6.5.3, “Wake Up from Stop or Wait Mode” for details.
- Freeze mode (BDM active)

  In freeze mode (BDM active), the interrupt vector base register is overridden internally. Please refer to Section 6.3.2.1, “Interrupt Vector Base Register (IVBR)” for details.

### 6.1.4 Block Diagram

Figure 6-1 shows a block diagram of the XINT module.

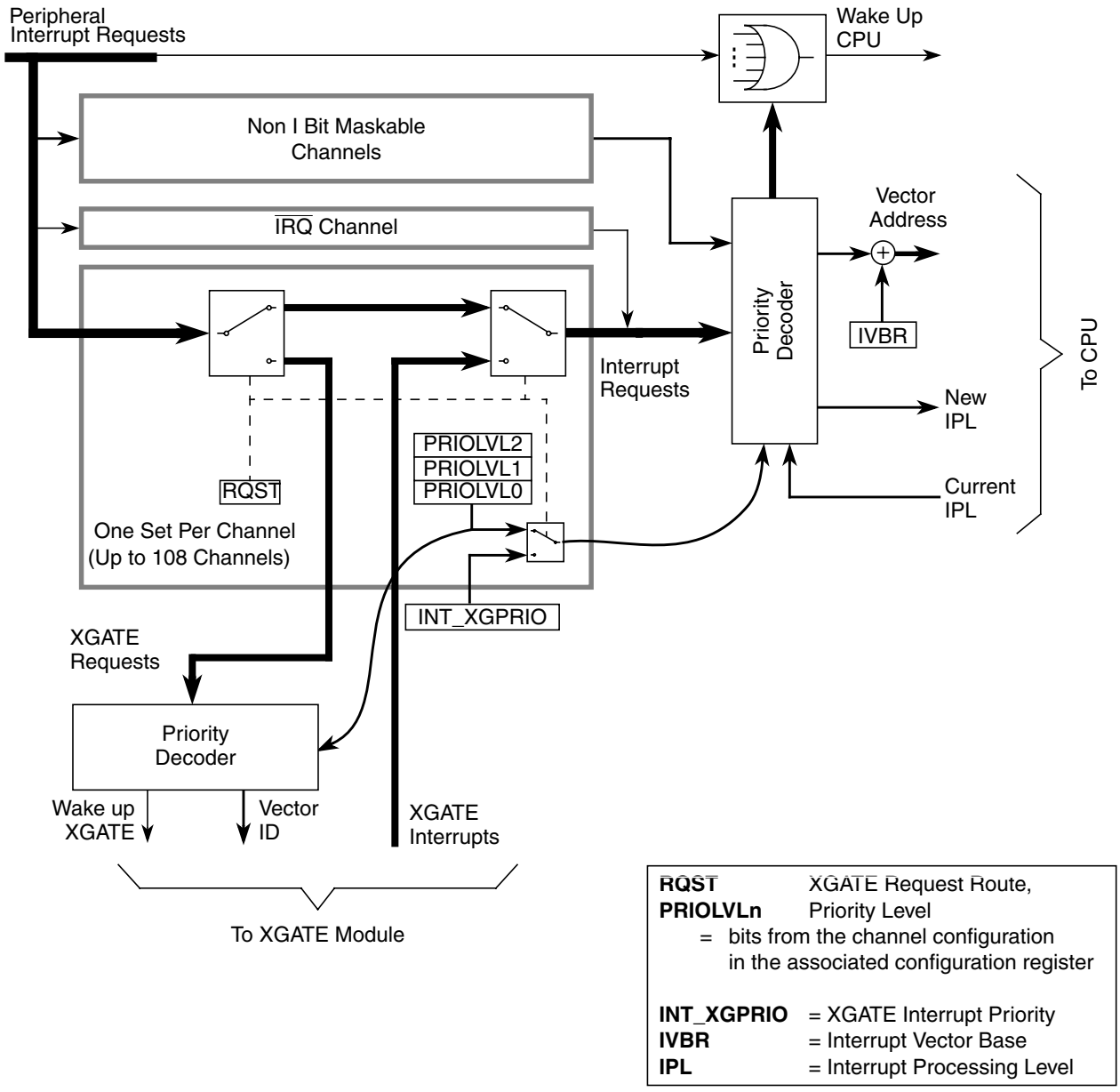



**Figure 6-1. XINT Block Diagram**

## 6.2 External Signal Description

The XINT module has no external signals.

## 6.3 Memory Map and Register Definition

This section provides a detailed description of all registers accessible in the XINT module.

### 6.3.1 Module Memory Map

Table 6-3 gives an overview over all XINT module registers.

**Table 6-3. XINT Memory Map**

| Address | Use | Access |
|---|---|---|
| 0x0120 | RESERVED | — |
| 0x0121 | Interrupt Vector Base Register (IVBR) | R/W |
| 0x0122–0x0125 | RESERVED | — |
| 0x0126 | XGATE Interrupt Priority Configuration Register (INT_XGPRIO) | R/W |
| 0x0127 | Interrupt Request Configuration Address Register (INT_CFADDR) | R/W |
| 0x0128 | Interrupt Request Configuration Data Register 0 (INT_CFDATA0) | R/W |
| 0x0129 | Interrupt Request Configuration Data Register 1 (INT_CFDATA1) | R/W |
| 0x012A | Interrupt Request Configuration Data Register 2 (INT_CFDATA2 | R/W |
| 0x012B | Interrupt Request Configuration Data Register 3 (INT_CFDATA3) | R/W |
| 0x012C | Interrupt Request Configuration Data Register 4 (INT_CFDATA4) | R/W |
| 0x012D | Interrupt Request Configuration Data Register 5 (INT_CFDATA5) | R/W |
| 0x012E | Interrupt Request Configuration Data Register 6 (INT_CFDATA6) | R/W |
| 0x012F | Interrupt Request Configuration Data Register 7 (INT_CFDATA7) | R/W |

## 6.3.2 Register Descriptions

This section describes in address order all the XINT module registers and their individual bits.

| Address | Register Name | | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|---|---|
| 0x0121 | IVBR | R<br>W | IVB_ADDR[7:0]7 | | | | | | | |
| 0x0126 | INT_XGPRIO | R<br>W | 0 | 0 | 0 | 0 | 0 | XILVL[2:0] | | |
| 0x0127 | INT_CFADDR | R<br>W | INT_CFADDR[7:4] | | | | 0 | 0 | 0 | 0 |
| 0x0128 | INT_CFDATA0 | R<br>W | RQST | 0 | 0 | 0 | 0 | PRIOLVL[2:0] | | |
| 0x0129 | INT_CFDATA1 | R<br>W | RQST | 0 | 0 | 0 | 0 | PRIOLVL[2:0] | | |
| 0x012A | INT_CFDATA2 | R<br>W | RQST | 0 | 0 | 0 | 0 | PRIOLVL[2:0] | | |
| 0x012B | INT_CFDATA3 | R<br>W | RQST | 0 | 0 | 0 | 0 | PRIOLVL[2:0] | | |
| 0x012C | INT_CFDATA4 | R<br>W | RQST | 0 | 0 | 0 | 0 | PRIOLVL[2:0] | | |
| 0x012D | INT_CFDATA5 | R<br>W | RQST | 0 | 0 | 0 | 0 | PRIOLVL[2:0] | | |
| 0x012E | INT_CFDATA6 | R<br>W | RQST | 0 | 0 | 0 | 0 | PRIOLVL[2:0] | | |
| 0x012F | INT_CFDATA7 | R<br>W | RQST | 0 | 0 | 0 | 0 | PRIOLVL[2:0] | | |

= Unimplemented or Reserved

**Figure 6-2. XINT Register Summary**

### 6.3.2.1 Interrupt Vector Base Register (IVBR)

Address: 0x0121

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | IVB_ADDR[7:0] | | | | | | | |
| Reset | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

**Figure 6-3. Interrupt Vector Base Register (IVBR)**

Read: Anytime

Write: Anytime

**Table 6-4. IVBR Field Descriptions**

| Field | Description |
|---|---|
| 7–0<br>IVB_ADDR[7:0] | **Interrupt Vector Base Address Bits** — These bits represent the upper byte of all vector addresses. Out of reset these bits are set to 0xFF (i.e., vectors are located at 0xFF10–0xFFFE) to ensure compatibility to previous S12 microcontrollers.<br>**Note:** A system reset will initialize the interrupt vector base register with “0xFF” before it is used to determine the reset vector address. Therefore, changing the IVBR has no effect on the location of the three reset vectors (0xFFFA–0xFFFE).<br>**Note:** If the BDM is active (i.e., the CPU is in the process of executing BDM firmware code), the contents of IVBR are ignored and the upper byte of the vector address is fixed as “0xFF”. |

### 6.3.2.2 XGATE Interrupt Priority Configuration Register (INT_XGPRIO)

Address: 0x0126

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | 0 | 0 | 0 | XILVL[2:0] | | |
| W | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |

= Unimplemented or Reserved

**Figure 6-4. XGATE Interrupt Priority Configuration Register (INT_XGPRIO)**

Read: Anytime

Write: Anytime

**Table 6-5. INT_XGPRIO Field Descriptions**

| Field | Description |
|---|---|
| 2–0<br>XILVL[2:0] | **XGATE Interrupt Priority Level** — The XILVL[2:0] bits configure the shared interrupt level of the XGATE interrupts coming from the XGATE module. Out of reset the priority is set to the lowest active level (“1”).<br>**Note:** If the XGATE module is not available on the device, write accesses to this register are ignored and read accesses to this register will return all 0. |

**Table 6-6. XGATE Interrupt Priority Levels**

| Priority | XILVL2 | XILVL1 | XILVL0 | Meaning |
|---|---|---|---|---|
| | 0 | 0 | 0 | Interrupt request is disabled |
| low | 0 | 0 | 1 | Priority level 1 |
| | 0 | 1 | 0 | Priority level 2 |
| | 0 | 1 | 1 | Priority level 3 |
| | 1 | 0 | 0 | Priority level 4 |
| | 1 | 0 | 1 | Priority level 5 |
| | 1 | 1 | 0 | Priority level 6 |
| high | 1 | 1 | 1 | Priority level 7 |

### 6.3.2.3 Interrupt Request Configuration Address Register (INT_CFADDR)

Address: 0x0127

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | INT_CFADDR[7:4] | | | | 0 | 0 | 0 | 0 |
| W | | | | | | | | |
| Reset | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |

= Unimplemented or Reserved

**Figure 6-5. Interrupt Configuration Address Register (INT_CFADDR)**

Read: Anytime

Write: Anytime

**Table 6-7. INT_CFADDR Field Descriptions**

| Field | Description |
|---|---|
| 7–4<br>INT_CFADDR[7:4] | **Interrupt Request Configuration Data Register Select Bits** — These bits determine which of the 128 configuration data registers are accessible in the 8 register window at INT_CFDATA0–7. The hexadecimal value written to this register corresponds to the upper nibble of the lower byte of the address of the interrupt vector, i.e., writing 0xE0 to this register selects the configuration data register block for the 8 interrupt vector requests starting with vector at address (vector base + 0x00E0) to be accessible as INT_CFDATA0–7.<br>**Note:** Writing all 0s selects non-existing configuration registers. In this case write accesses to INT_CFDATA0–7 will be ignored and read accesses will return all 0. |

### 6.3.2.4 Interrupt Request Configuration Data Registers (INT_CFDATA0–7)

The eight register window visible at addresses INT_CFDATA0–7 contains the configuration data for the block of eight interrupt requests (out of 128) selected by the interrupt configuration address register (INT_CFADDR) in ascending order. INT_CFDATA0 represents the interrupt configuration data register of the vector with the lowest address in this block, while INT_CFDATA7 represents the interrupt configuration data register of the vector with the highest address, respectively.

Address: 0x0128

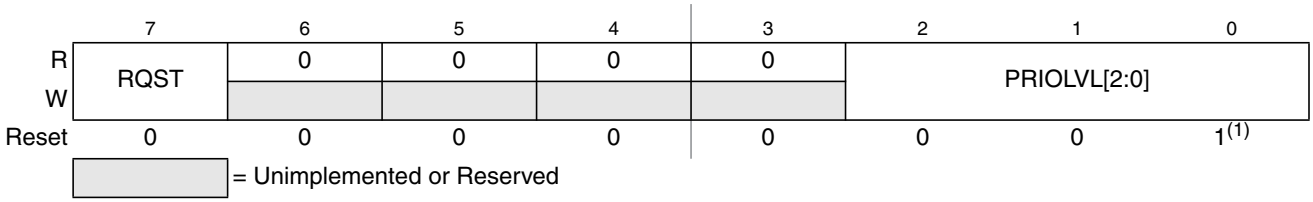



**Figure 6-6. Interrupt Request Configuration Data Register 0 (INT_CFDATA0)**

1. Please refer to the notes following the PRIOLVL[2:0] description below.

Address: 0x0129

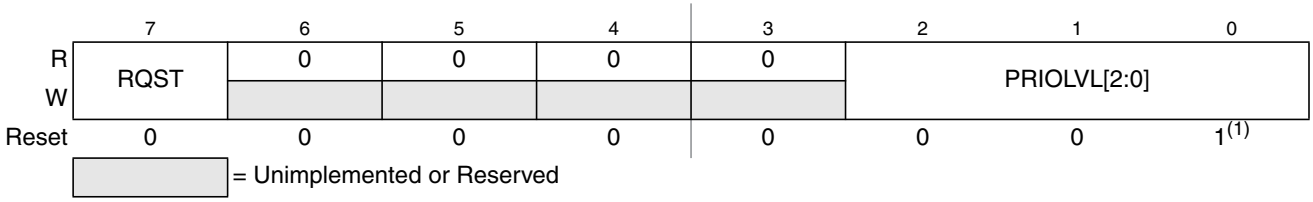



**Figure 6-7. Interrupt Request Configuration Data Register 1 (INT_CFDATA1)**

1. Please refer to the notes following the PRIOLVL[2:0] description below.

Address: 0x012A

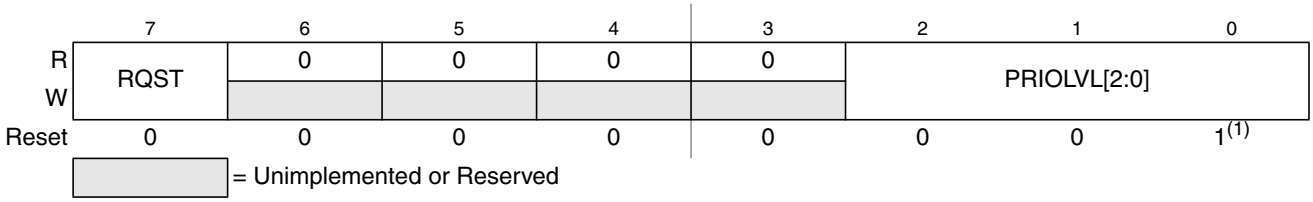



**Figure 6-8. Interrupt Request Configuration Data Register 2 (INT_CFDATA2)**

1. Please refer to the notes following the PRIOLVL[2:0] description below.

Address: 0x012B

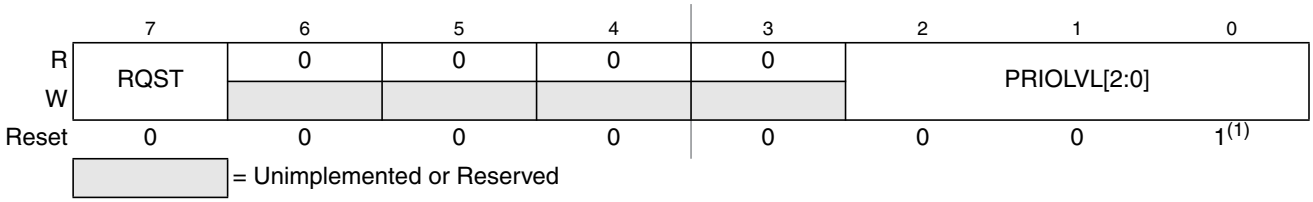



**Figure 6-9. Interrupt Request Configuration Data Register 3 (INT_CFDATA3)**

1. Please refer to the notes following the PRIOLVL[2:0] description below.

Address: 0x012C

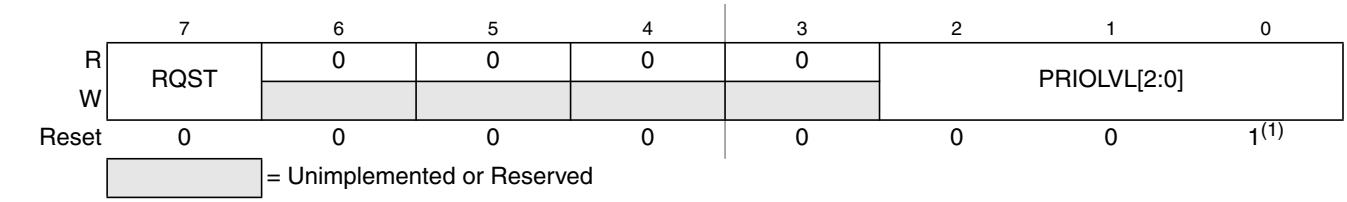



**Figure 6-10. Interrupt Request Configuration Data Register 4 (INT_CFDATA4)**

1. Please refer to the notes following the PRIOLVL[2:0] description below.

Address: 0x012D

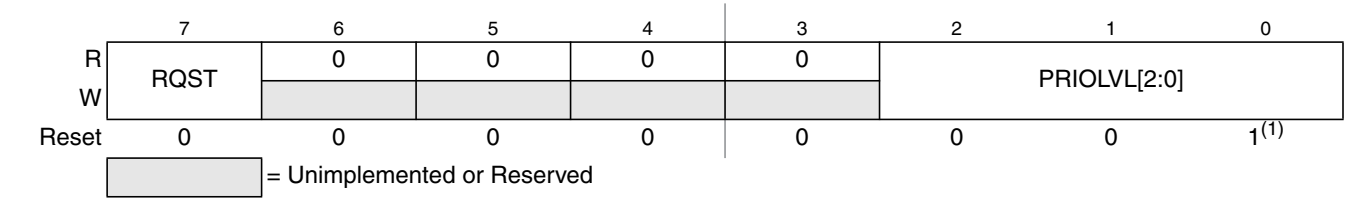



**Figure 6-11. Interrupt Request Configuration Data Register 5 (INT_CFDATA5)**

1. Please refer to the notes following the PRIOLVL[2:0] description below.

Address: 0x012E

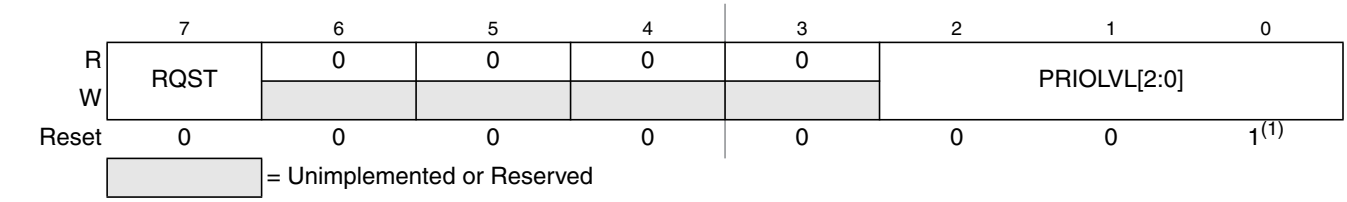



**Figure 6-12. Interrupt Request Configuration Data Register 6 (INT_CFDATA6)**

1. Please refer to the notes following the PRIOLVL[2:0] description below.

Address: 0x012F

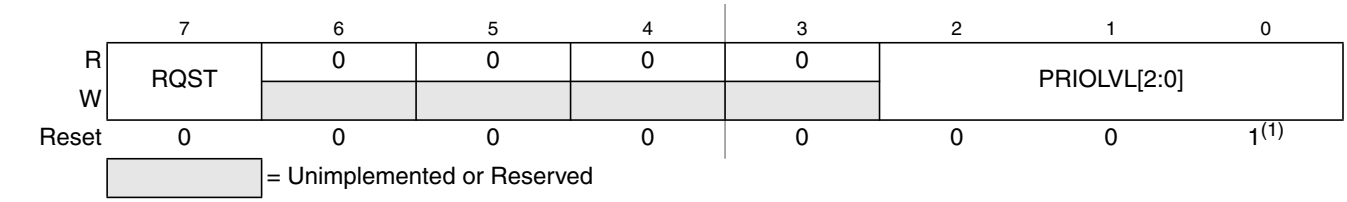



**Figure 6-13. Interrupt Request Configuration Data Register 7 (INT_CFDATA7)**

1. Please refer to the notes following the PRIOLVL[2:0] description below.

Read: Anytime

Write: Anytime

**Table 6-8. INT_CFDATA0–7 Field Descriptions**

| Field | Description |
|---|---|
| 7<br>RQST | **XGATE Request Enable** — This bit determines if the associated interrupt request is handled by the CPU or by the XGATE module.<br>0 Interrupt request is handled by the CPU<br>1 Interrupt request is handled by the XGATE module<br>**Note:** The $\overline{IRQ}$ interrupt cannot be handled by the XGATE module. For this reason, the configuration register for vector (vector base + 0x00F2) = $\overline{IRQ}$ vector address) does not contain a RQST bit. Writing a 1 to the location of the RQST bit in this register will be ignored and a read access will return 0.<br>**Note:** If the XGATE module is not available on the device, writing a 1 to the location of the RQST bit in this register will be ignored and a read access will return 0. |
| 2–0<br>PRIOLVL[2:0] | **Interrupt Request Priority Level Bits** — The PRIOLVL[2:0] bits configure the interrupt request priority level of the associated interrupt request. Out of reset all interrupt requests are enabled at the lowest active level (“1”) to provide backwards compatibility with previous S12 interrupt controllers. Please also refer to Table 6-9 for available interrupt request priority levels.<br>**Note:** Write accesses to configuration data registers of unused interrupt channels will be ignored and read accesses will return all 0. For information about what interrupt channels are used in a specific MCU, please refer to the Device Reference Manual of that MCU.<br>**Note:** When vectors (vector base + 0x00F0–0x00FE) are selected by writing 0xF0 to INT_CFADDR, writes to INT_CFDATA2–7 (0x00F4–0x00FE) will be ignored and read accesses will return all 0s. The corresponding vectors do not have configuration data registers associated with them.<br>**Note:** When vectors (vector base + 0x0010–0x001E) are selected by writing 0x10 to INT_CFADDR, writes to INT_CFDATA1–INT_CFDATA4 (0x0012–0x0018) will be ignored and read accesses will return all 0s. The corresponding vectors do not have configuration data registers associated with them.<br>**Note:** Write accesses to the configuration register for the spurious interrupt vector request (vector base + 0x0010) will be ignored and read accesses will return 0x07 (request is handled by the CPU, PRIOLVL = 7). |

**Table 6-9. Interrupt Priority Levels**

| Priority | PRIOLVL2 | PRIOLVL1 | PRIOLVL0 | Meaning |
|---|---|---|---|---|
| | 0 | 0 | 0 | Interrupt request is disabled |
| low | 0 | 0 | 1 | Priority level 1 |
| | 0 | 1 | 0 | Priority level 2 |
| | 0 | 1 | 1 | Priority level 3 |
| | 1 | 0 | 0 | Priority level 4 |
| | 1 | 0 | 1 | Priority level 5 |
| | 1 | 1 | 0 | Priority level 6 |
| high | 1 | 1 | 1 | Priority level 7 |

## 6.4 Functional Description

The XINT module processes all exception requests to be serviced by the CPU module. These exceptions include interrupt vector requests and reset vector requests. Each of these exception types and their overall priority level is discussed in the subsections below.

### 6.4.1 S12X Exception Requests

The CPU handles both reset requests and interrupt requests. The XINT module contains registers to configure the priority level of each I bit maskable interrupt request which can be used to implement an interrupt priority scheme. This also includes the possibility to nest interrupt requests. A priority decoder is used to evaluate the priority of a pending interrupt request.

### 6.4.2 Interrupt Prioritization

After system reset all interrupt requests with a vector address lower than or equal to (vector base + 0x00F2) are enabled, are set up to be handled by the CPU and have a pre-configured priority level of 1. Exceptions to this rule are the non-maskable interrupt requests and the spurious interrupt vector request at (vector base + 0x0010) which cannot be disabled, are always handled by the CPU and have a fixed priority levels. A priority level of 0 effectively disables the associated I bit maskable interrupt request.

If more than one interrupt request is configured to the same interrupt priority level the interrupt request with the higher vector address wins the prioritization.

The following conditions must be met for an I bit maskable interrupt request to be processed.

1. The local interrupt enabled bit in the peripheral module must be set.
2. The setup in the configuration register associated with the interrupt request channel must meet the following conditions:
   a) The XGATE request enable bit must be 0 to have the CPU handle the interrupt request.
   b) The priority level must be set to non zero.
   c) The priority level must be greater than the current interrupt processing level in the condition code register (CCR) of the CPU (PRIOLVL[2:0] > IPL[2:0]).
3. The I bit in the condition code register (CCR) of the CPU must be cleared.
4. There is no access violation interrupt request pending.
5. There is no SYS, SWI, BDM, TRAP, or $\overline{\text{XIRQ}}$ request pending.

**NOTE**

All non I bit maskable interrupt requests always have higher priority than I bit maskable interrupt requests. If an I bit maskable interrupt request is interrupted by a non I bit maskable interrupt request, the currently active interrupt processing level (IPL) remains unaffected. It is possible to nest non I bit maskable interrupt requests, e.g., by nesting SWI or TRAP calls.

#### 6.4.2.1 Interrupt Priority Stack

The current interrupt processing level (IPL) is stored in the condition code register (CCR) of the CPU. This way the current IPL is automatically pushed to the stack by the standard interrupt stacking procedure. The new IPL is copied to the CCR from the priority level of the highest priority active interrupt request channel which is configured to be handled by the CPU. The copying takes place when the interrupt vector is fetched. The previous IPL is automatically restored by executing the RTI instruction.

### 6.4.3 XGATE Requests

If the XGATE module is implemented on the device, the XINT module is also used to process all exception requests to be serviced by the XGATE module. The overall priority level of those exceptions is discussed in the subsections below.

#### 6.4.3.1 XGATE Request Prioritization

An interrupt request channel is configured to be handled by the XGATE module, if the RQST bit of the associated configuration register is set to 1 (please refer to Section 6.3.2.4, “Interrupt Request Configuration Data Registers (INT_CFDATA0–7)”). The priority level configuration (PRIOLVL) for this channel becomes the XGATE priority which will be used to determine the highest priority XGATE request to be serviced next by the XGATE module. Additionally, XGATE interrupts may be raised by the XGATE module by setting one or more of the XGATE channel interrupt flags (by using the SIF instruction). This will result in an CPU interrupt with vector address vector base + (2 * channel ID number), where the channel ID number corresponds to the highest set channel interrupt flag, if the XGIE and channel RQST bits are set.

The shared interrupt priority for the XGATE interrupt requests is taken from the XGATE interrupt priority configuration register (please refer to Section 6.3.2.2, “XGATE Interrupt Priority Configuration Register (INT_XGPRIO)”). If more than one XGATE interrupt request channel becomes active at the same time, the channel with the highest vector address wins the prioritization.

### 6.4.4 Priority Decoders

The XINT module contains priority decoders to determine the priority for all interrupt requests pending for the respective target.

There are two priority decoders, one for each interrupt request target, CPU or XGATE. The function of both priority decoders is basically the same with one exception: the priority decoder for the XGATE module does not take the current XGATE thread processing level into account. Instead, XGATE requests are handed to the XGATE module including a 1-bit priority identifier. The XGATE module uses this additional information to decide if the new request can interrupt a currently running thread. The 1-bit priority identifier corresponds to the most significant bit of the priority level configuration of the requesting channel. This means that XGATE requests with priority levels 4, 5, 6 or 7 can interrupt running XGATE threads with priority levels 1, 2 and 3.

A CPU interrupt vector is not supplied until the CPU requests it. Therefore, it is possible that a higher priority interrupt request could override the original exception which caused the CPU to request the vector. In this case, the CPU will receive the highest priority vector and the system will process this exception instead of the original request.

If the interrupt source is unknown (for example, in the case where an interrupt request becomes inactive after the interrupt has been recognized, but prior to the vector request), the vector address supplied to the CPU will default to that of the spurious interrupt vector.

**NOTE**

Care must be taken to ensure that all exception requests remain active until the system begins execution of the applicable service routine; otherwise, the exception request may not get processed at all or the result may be a spurious interrupt request (vector at address (vector base + 0x0010)).

## 6.4.5 Reset Exception Requests

The XINT module supports three system reset exception request types (for details please refer to the Clock and Reset Generator module (CRG)):

1. Pin reset, power-on reset, low-voltage reset, or illegal address reset
2. Clock monitor reset request
3. COP watchdog reset request

## 6.4.6 Exception Priority

The priority (from highest to lowest) and address of all exception vectors issued by the XINT module upon request by the CPU is shown in Table 6-10. Generally, all non-maskable interrupts have higher priorities than maskable interrupts. Please note that between the three software interrupts (Unimplemented op-code trap request, SWI/BGND request, SYS request) there is no real priority defined because they cannot occur simultaneously (the S12XCPU executes one instruction at a time).

**Table 6-10. Exception Vector Map and Priority**

| Vector Address[1] | Source |
|---|---|
| 0xFFFE | Pin reset, power-on reset, low-voltage reset, illegal address reset |
| 0xFFFC | Clock monitor reset |
| 0xFFFA | COP watchdog reset |
| (Vector base + 0x00F8) | Unimplemented op-code trap |
| (Vector base + 0x00F6) | Software interrupt instruction (SWI) or BDM vector request |
| (Vector base + 0x0012) | System call interrupt instruction (SYS) |
| (Vector base + 0x0018) | (reserved for future use) |
| (Vector base + 0x0016) | XGATE Access violation interrupt request[2] |
| (Vector base + 0x0014) | CPU Access violation interrupt request[3] |
| (Vector base + 0x00F4) | $\overline{\text{XIRQ}}$ interrupt request |
| (Vector base + 0x00F2) | $\overline{\text{IRQ}}$ interrupt request |
| (Vector base + 0x00F0–0x001A) | Device specific I bit maskable interrupt sources (priority determined by the associated configuration registers, in descending order) |
| (Vector base + 0x0010) | Spurious interrupt |

1. 16 bits vector address based
2. only implemented if device features both a Memory Protection Unit (MPU) and an XGATE co-processor
3. only implemented if device features a Memory Protection Unit (MPU)

## 6.5 Initialization/Application Information

### 6.5.1 Initialization

After system reset, software should:

- Initialize the interrupt vector base register if the interrupt vector table is not located at the default location (0xFF10–0xFFF9).
- Initialize the interrupt processing level configuration data registers (INT_CFADDR, INT_CFDATA0–7) for all interrupt vector requests with the desired priority levels and the request target (CPU or XGATE module). It might be a good idea to disable unused interrupt requests.
- If the XGATE module is used, setup the XGATE interrupt priority register (INT_XGPRIO) and configure the XGATE module (please refer the XGATE Block Guide for details).
- Enable I maskable interrupts by clearing the I bit in the CCR.
- Enable the X maskable interrupt by clearing the X bit in the CCR (if required).

### 6.5.2 Interrupt Nesting

The interrupt request priority level scheme makes it possible to implement priority based interrupt request nesting for the I bit maskable interrupt requests handled by the CPU.

- I bit maskable interrupt requests can be interrupted by an interrupt request with a higher priority, so that there can be up to seven nested I bit maskable interrupt requests at a time (refer to Figure 6-14 for an example using up to three nested interrupt requests).

I bit maskable interrupt requests cannot be interrupted by other I bit maskable interrupt requests per default. In order to make an interrupt service routine (ISR) interruptible, the ISR must explicitly clear the I bit in the CCR (CLI). After clearing the I bit, I bit maskable interrupt requests with higher priority can interrupt the current ISR.

An ISR of an interruptible I bit maskable interrupt request could basically look like this:

- Service interrupt, e.g., clear interrupt flags, copy data, etc.
- Clear I bit in the CCR by executing the instruction CLI (thus allowing interrupt requests with higher priority)
- Process data
- Return from interrupt by executing the instruction RTI

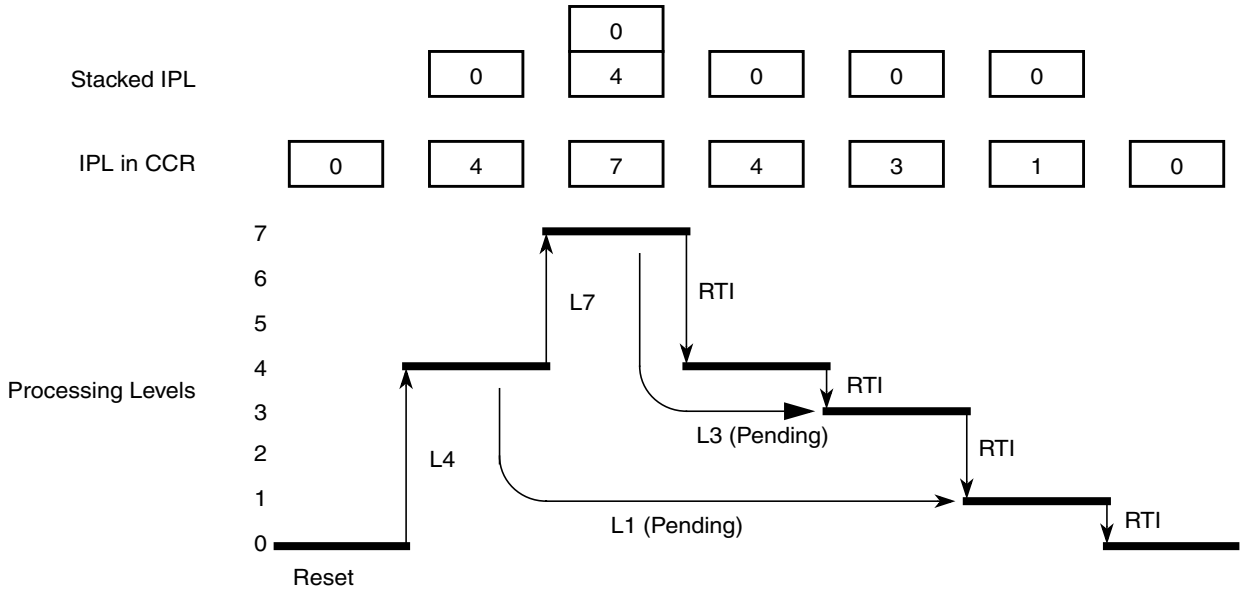



**Figure 6-14. Interrupt Processing Example**

## 6.5.3 Wake Up from Stop or Wait Mode

### 6.5.3.1 CPU Wake Up from Stop or Wait Mode

Only I bit maskable interrupt requests which are configured to be handled by the CPU are capable of waking the MCU from wait mode.

Since bus and core clocks are disabled in stop mode, only interrupt requests that can be generated without these clocks can wake the MCU from stop mode. These are listed in the device overview interrupt vector table. Only I bit maskable interrupt requests which are configured to be handled by the CPU are capable of waking the MCU from stop mode.

To determine whether an I bit maskable interrupt is qualified to wake up the CPU or not, the same settings as in normal run mode are applied during stop or wait mode:

- If the I bit in the CCR is set, all I bit maskable interrupts are masked from waking up the MCU.
- An I bit maskable interrupt is ignored if it is configured to a priority level below or equal to the current IPL in CCR.
- I bit maskable interrupt requests which are configured to be handled by the XGATE module are not capable of waking up the CPU.

The X bit maskable interrupt request can wake up the MCU from stop or wait mode at anytime, even if the X bit in CCR is set. If the X bit maskable interrupt request is used to wake-up the MCU with the X bit in the CCR set, the associated ISR is not called. The CPU then resumes program execution with the instruction following the WAI or STOP instruction. This features works following the same rules like any interrupt request, i.e. care must be taken that the X interrupt request used for wake-up remains active at least until the system begins execution of the instruction following the WAI or STOP instruction; otherwise, wake-up may not occur.

#### 6.5.3.2 XGATE Wake Up from Stop or Wait Mode

Interrupt request channels which are configured to be handled by the XGATE module are capable of waking up the XGATE module. Interrupt request channels handled by the XGATE module do not affect the state of the CPU.

# Chapter 7
# Background Debug Module (S12XBDMV2)

**Table 7-1. Revision History**

| Revision Number | Revision Date | Sections Affected | Description of Changes |
|---|---|---|---|
| V02.00 | 07 Mar 2006 | | - First version of S12XBDMV2 |
| V02.01 | 14 May 2008 | | - Introduced standardized Revision History Table |
| V02.02 | 12 Sep 2012 | | - Minor formatting corrections |

## 7.1 Introduction

This section describes the functionality of the background debug module (BDM) sub-block of the HCS12X core platform.

The background debug module (BDM) sub-block is a single-wire, background debug system implemented in on-chip hardware for minimal CPU intervention. All interfacing with the BDM is done via the BKGD pin.

The BDM has enhanced capability for maintaining synchronization between the target and host while allowing more flexibility in clock rates. This includes a sync signal to determine the communication rate and a handshake signal to indicate when an operation is complete. The system is backwards compatible to the BDM of the S12 family with the following exceptions:

- TAGGO command no longer supported by BDM
- External instruction tagging feature now part of DBG module
- BDM register map and register content extended/modified
- Global page access functionality
- Enabled but not active out of reset in emulation modes (if modes available)
- CLKSW bit set out of reset in emulation modes (if modes available).
- Family ID readable from firmware ROM at global address 0x7FFF0F (value for HCS12X devices is 0xC1)

### 7.1.1 Features

The BDM includes these distinctive features:

- Single-wire communication with host development system
- Enhanced capability for allowing more flexibility in clock rates
- SYNC command to determine communication rate
- GO_UNTIL command

- Hardware handshake protocol to increase the performance of the serial communication
- Active out of reset in special single chip mode
- Nine hardware commands using free cycles, if available, for minimal CPU intervention
- Hardware commands not requiring active BDM
- 14 firmware commands execute from the standard BDM firmware lookup table
- Software control of BDM operation during wait mode
- Software selectable clocks
- Global page access functionality
- Enabled but not active out of reset in emulation modes (if modes available)
- CLKSW bit set out of reset in emulation modes (if modes available).
- When secured, hardware commands are allowed to access the register space in special single chip mode, if the non-volatile memory erase test fail.
- Family ID readable from firmware ROM at global address 0x7FFF0F (value for HCS12X devices is 0xC1)
- BDM hardware commands are operational until system stop mode is entered (all bus masters are in stop mode)

## 7.1.2 Modes of Operation

BDM is available in all operating modes but must be enabled before firmware commands are executed. Some systems may have a control bit that allows suspending thefunction during background debug mode.

### 7.1.2.1 Regular Run Modes

All of these operations refer to the part in run mode and not being secured. The BDM does not provide controls to conserve power during run mode.

- Normal modes

  General operation of the BDM is available and operates the same in all normal modes.
- Special single chip mode

  In special single chip mode, background operation is enabled and active out of reset. This allows programming a system with blank memory.
- Emulation modes (if modes available)

  In emulation mode, background operation is enabled but not active out of reset. This allows debugging and programming a system in this mode more easily.

### 7.1.2.2 Secure Mode Operation

If the device is in secure mode, the operation of the BDM is reduced to a small subset of its regular run mode operation. Secure operation prevents BDM and CPU accesses to non-volatile memory (Flash and/or EEPROM) other than allowing erasure. For more information please see Section 7.4.1, “Security”.

#### 7.1.2.3 Low-Power Modes

The BDM can be used until all bus masters (e.g., CPU or XGATE or others depending on which masters are available on the SOC) are in stop mode. When CPU is in a low power mode (wait or stop mode) all BDM firmware commands as well as the hardware BACKGROUND command can not be used respectively are ignored. In this case the CPU can not enter BDM active mode, and only hardware read and write commands are available. Also the CPU can not enter a low power mode during BDM active mode.

If all bus masters are in stop mode, the BDM clocks are stopped as well. When BDM clocks are disabled and one of the bus masters exits from stop mode the BDM clocks will restart and BDM will have a soft reset (clearing the instruction register, any command in progress and disable the ACK function). The BDM is now ready to receive a new command.

### 7.1.3 Block Diagram

A block diagram of the BDM is shown in Figure 7-1.

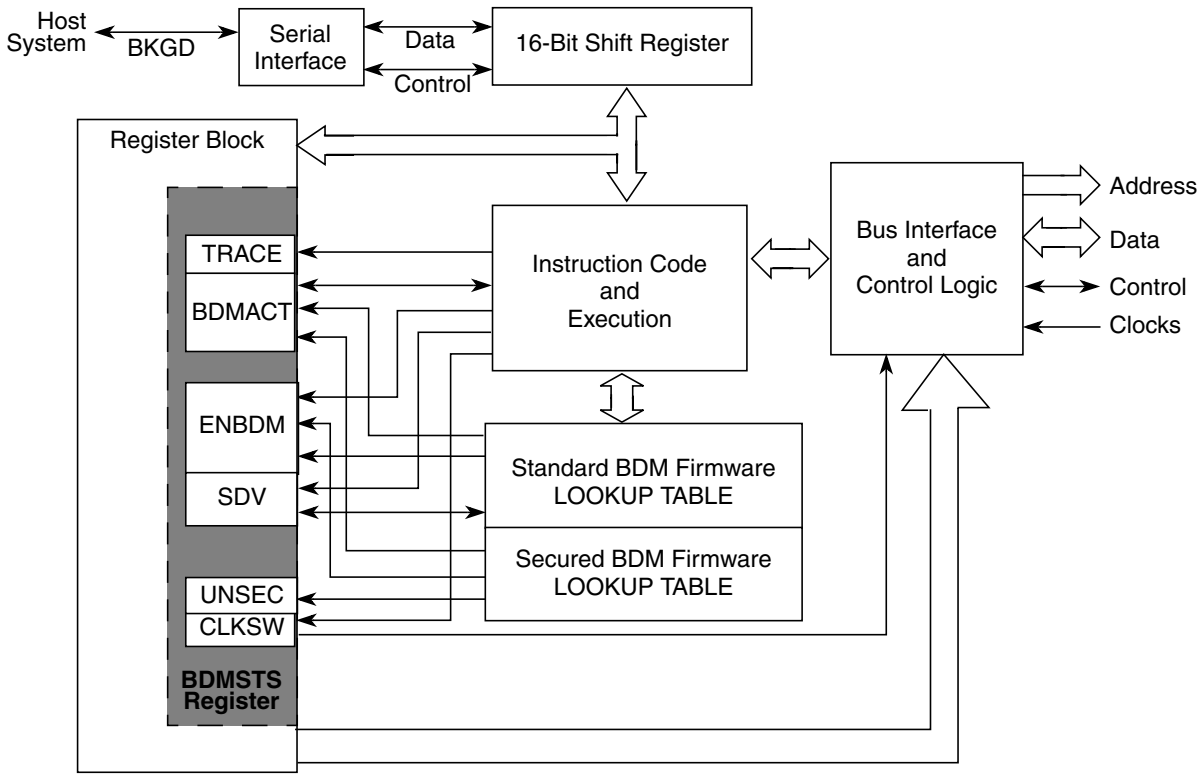



**Figure 7-1. BDM Block Diagram**

## 7.2 External Signal Description

A single-wire interface pin called the background debug interface (BKGD) pin is used to communicate with the BDM system. During reset, this pin is a mode select input which selects between normal and special modes of operation. After reset, this pin becomes the dedicated serial interface pin for the background debug mode.

## 7.3 Memory Map and Register Definition

### 7.3.1 Module Memory Map

Table 7-2 shows the BDM memory map when BDM is active.

**Table 7-2. BDM Memory Map**

| Global Address | Module | Size (Bytes) |
|---|---|---|
| 0x7FFF00–0x7FFF0B | BDM registers | 12 |
| 0x7FFF0C–0x7FFF0E | BDM firmware ROM | 3 |
| 0x7FFF0F | Family ID (part of BDM firmware ROM) | 1 |
| 0x7FFF10–0x7FFFFF | BDM firmware ROM | 240 |

### 7.3.2 Register Descriptions

A summary of the registers associated with the BDM is shown in Figure 7-2. Registers are accessed by host-driven communications to the BDM hardware using READ_BD and WRITE_BD commands.

| Global Address | Register Name | | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|---|---|
| 0x7FFF00 | Reserved | R | X | X | X | X | X | X | 0 | 0 |
| | | W | | | | | | | | |
| 0x7FFF01 | BDMSTS | R | ENBDM | BDMACT | 0 | SDV | TRACE | CLKSW | UNSEC | 0 |
| | | W | | | | | | | | |
| 0x7FFF02 | Reserved | R | X | X | X | X | X | X | X | X |
| | | W | | | | | | | | |
| 0x7FFF03 | Reserved | R | X | X | X | X | X | X | X | X |
| | | W | | | | | | | | |
| 0x7FFF04 | Reserved | R | X | X | X | X | X | X | X | X |
| | | W | | | | | | | | |
| 0x7FFF05 | Reserved | R | X | X | X | X | X | X | X | X |
| | | W | | | | | | | | |
| 0x7FFF06 | BDMCCRL | R | CCR7 | CCR6 | CCR5 | CCR4 | CCR3 | CCR2 | CCR1 | CCR0 |
| | | W | | | | | | | | |

= Unimplemented, Reserved; = Implemented (do not alter)

X = Indeterminate; 0 = Always read zero

**Figure 7-2. BDM Register Summary**

| Global Address | Register Name | | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|---|---|
| 0x7FFF07 | BDMCCRH | R | 0 | 0 | 0 | 0 | 0 | CCR10 | CCR9 | CCR8 |
| | | W | | | | | | | | |
| 0x7FFF08 | BDMGPR | R | BGAE | BGP6 | BGP5 | BGP4 | BGP3 | BGP2 | BGP1 | BGP0 |
| | | W | | | | | | | | |
| 0x7FFF09 | Reserved | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | W | | | | | | | | |
| 0x7FFF0A | Reserved | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | W | | | | | | | | |
| 0x7FFF0B | Reserved | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | W | | | | | | | | |

= Unimplemented, Reserved; = Implemented (do not alter)

X = Indeterminate; 0 = Always read zero

**Figure 7-2. BDM Register Summary (continued)**

## 7.3.2.1 BDM Status Register (BDMSTS)

Register Global Address 0x7FFF01

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | ENBDM | BDMACT | 0 | SDV | TRACE | CLKSW | UNSEC | 0 |
| W | | | | | | | | |
| Reset | | | | | | | | |
| Special Single-Chip Mode | 0(1) | 1 | 0 | 0 | 0 | 0 | 0(3) | 0 |
| Emulation Modes (if modes available) | 1 | 0 | 0 | 0 | 0 | 1(2) | 0 | 0 |
| All Other Modes | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented, Reserved; = Implemented (do not alter)

0 = Always read zero

1. ENBDM is read as 1 by a debugging environment in special single chip mode when the device is not secured or secured but fully erased (non-volatile memory). This is because the ENBDM bit is set by the standard firmware before a BDM command can be fully transmitted and executed.
2. CLKSW is read as 1 by a debugging environment in emulation modes when the device is not secured and read as 0 when secured if emulation modes available.
3. UNSEC is read as 1 by a debugging environment in special single chip mode when the device is secured and fully erased, else it is 0 and can only be read if not secure (see also bit description).

**Figure 7-3. BDM Status Register (BDMSTS)**

Read: All modes through BDM operation when not secured

Write: All modes through BDM operation when not secured, but subject to the following:

— ENBDM should only be set via a BDM hardware command if the BDM firmware commands are needed. (This does not apply in special single chip and emulation modes).

— BDMACT can only be set by BDM hardware upon entry into BDM. It can only be cleared by the standard BDM firmware lookup table upon exit from BDM active mode.

— CLKSW can only be written via BDM hardware WRITE_BD commands.

— All other bits, while writable via BDM hardware or standard BDM firmware write commands, should only be altered by the BDM hardware or standard firmware lookup table as part of BDM command execution.

**Table 7-3. BDMSTS Field Descriptions**

| Field | Description |
|---|---|
| 7<br>ENBDM | **Enable BDM** — This bit controls whether the BDM is enabled or disabled. When enabled, BDM can be made active to allow firmware commands to be executed. When disabled, BDM cannot be made active but BDM hardware commands are still allowed.<br>0 BDM disabled<br>1 BDM enabled<br>**Note:** ENBDM is set by the firmware out of reset in special single chip mode. In emulation modes (if modes available) the ENBDM bit is set by BDM hardware out of reset. In special single chip mode with the device secured, this bit will not be set by the firmware until after the non-volatile memory erase verify tests are complete. In emulation modes (if modes available) with the device secured, the BDM operations are blocked. |
| 6<br>BDMACT | **BDM Active Status** — This bit becomes set upon entering BDM. The standard BDM firmware lookup table is then enabled and put into the memory map. BDMACT is cleared by a carefully timed store instruction in the standard BDM firmware as part of the exit sequence to return to user code and remove the BDM memory from the map.<br>0 BDM not active<br>1 BDM active |
| 4<br>SDV | **Shift Data Valid** — This bit is set and cleared by the BDM hardware. It is set after data has been transmitted as part of a firmware or hardware read command or after data has been received as part of a firmware or hardware write command. It is cleared when the next BDM command has been received or BDM is exited. SDV is used by the standard BDM firmware to control program flow execution.<br>0 Data phase of command not complete<br>1 Data phase of command is complete |
| 3<br>TRACE | **TRACE1 BDM Firmware Command is Being Executed** — This bit gets set when a BDM TRACE1 firmware command is first recognized. It will stay set until BDM firmware is exited by one of the following BDM commands: GO or GO_UNTIL.<br>0 TRACE1 command is not being executed<br>1 TRACE1 command is being executed |

**Table 7-3. BDMSTS Field Descriptions (continued)**

| Field | Description |
|---|---|
| 2<br>CLKSW | **Clock Switch** — The CLKSW bit controls which clock the BDM operates with. It is only writable from a hardware BDM command. A minimum delay of 150 cycles at the clock speed that is active during the data portion of the command send to change the clock source should occur before the next command can be send. The delay should be obtained no matter which bit is modified to effectively change the clock source (either PLLSEL bit or CLKSW bit). This guarantees that the start of the next BDM command uses the new clock for timing subsequent BDM communications.<br>Table 7-4 shows the resulting BDM clock source based on the CLKSW and the PLLSEL (PLL select in the CRG module, the bit is part of the CLKSEL register) bits.<br>**Note:** The BDM alternate clock source can only be selected when CLKSW = 0 and PLLSEL = 1. The BDM serial interface is now fully synchronized to the alternate clock source, when enabled. This eliminates frequency restriction on the alternate clock which was required on previous versions. Refer to the device specification to determine which clock connects to the alternate clock source input.<br>**Note:** If the acknowledge function is turned on, changing the CLKSW bit will cause the ACK to be at the new rate for the write command which changes it.<br>**Note:** In emulation modes (if modes available), the CLKSW bit will be set out of RESET. |
| 1<br>UNSEC | **Unsecure** — If the device is secured this bit is only writable in special single chip mode from the BDM secure firmware. It is in a zero state as secure mode is entered so that the secure BDM firmware lookup table is enabled and put into the memory map overlapping the standard BDM firmware lookup table.<br>The secure BDM firmware lookup table verifies that the non-volatile memories (e.g. on-chip EEPROM and/or Flash EEPROM) are erased. This being the case, the UNSEC bit is set and the BDM program jumps to the start of the standard BDM firmware lookup table and the secure BDM firmware lookup table is turned off. If the erase test fails, the UNSEC bit will not be asserted.<br>0 System is in a secured mode.<br>1 System is in a unsecured mode.<br>**Note:** When UNSEC is set, security is off and the user can change the state of the secure bits in the on-chip Flash EEPROM. Note that if the user does not change the state of the bits to “unsecured” mode, the system will be secured again when it is next taken out of reset.After reset this bit has no meaning or effect when the security byte in the Flash EEPROM is configured for unsecure mode. |

**Table 7-4. BDM Clock Sources**

| PLLSEL | CLKSW | BDMCLK |
|---|---|---|
| 0 | 0 | Bus clock dependent on oscillator |
| 0 | 1 | Bus clock dependent on oscillator |
| 1 | 0 | Alternate clock (refer to the device specification to determine the alternate clock source) |
| 1 | 1 | Bus clock dependent on the PLL |

### 7.3.2.2 BDM CCR LOW Holding Register (BDMCCRL)

Register Global Address 0x7FFF06

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | CCR7 | CCR6 | CCR5 | CCR4 | CCR3 | CCR2 | CCR1 | CCR0 |
| Reset | | | | | | | | |
| Special Single-Chip Mode | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 0 |
| All Other Modes | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 7-4. BDM CCR LOW Holding Register (BDMCCRL)**

Read: All modes through BDM operation when not secured

Write: All modes through BDM operation when not secured

**NOTE**

When BDM is made active, the CPU stores the content of its $CCR_L$ register in the BDMCCRL register. However, out of special single-chip reset, the BDMCCRL is set to 0xD8 and not 0xD0 which is the reset value of the $CCR_L$ register in this CPU mode. Out of reset in all other modes the BDMCCRL register is read zero.

When entering background debug mode, the BDM CCR LOW holding register is used to save the low byte of the condition code register of the user's program. It is also used for temporary storage in the standard BDM firmware mode. The BDM CCR LOW holding register can be written to modify the CCR value.

### 7.3.2.3 BDM CCR HIGH Holding Register (BDMCCRH)

Register Global Address 0x7FFF07

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | 0 | 0 | 0 | CCR10 | CCR9 | CCR8 |
| W | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented or Reserved

**Figure 7-5. BDM CCR HIGH Holding Register (BDMCCRH)**

Read: All modes through BDM operation when not secured

Write: All modes through BDM operation when not secured

When entering background debug mode, the BDM CCR HIGH holding register is used to save the high byte of the condition code register of the user's program. The BDM CCR HIGH holding register can be written to modify the CCR value.

#### 7.3.2.4 BDM Global Page Index Register (BDMGPR)

Register Global Address 0x7FFF08

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | BGAE | BGP6 | BGP5 | BGP4 | BGP3 | BGP2 | BGP1 | BGP0 |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 7-6. BDM Global Page Register (BDMGPR)**

Read: All modes through BDM operation when not secured

Write: All modes through BDM operation when not secured

**Table 7-5. BDMGPR Field Descriptions**

| Field | Description |
|---|---|
| 7<br>BGAE | **BDM Global Page Access Enable Bit** — BGAE enables global page access for BDM hardware and firmware read/write instructions The BDM hardware commands used to access the BDM registers (READ_BD_ and WRITE_BD_) can not be used for global accesses even if the BGAE bit is set.<br>0 BDM Global Access disabled<br>1 BDM Global Access enabled |
| 6–0<br>BGP[6:0] | **BDM Global Page Index Bits 6–0** — These bits define the extended address bits from 22 to 16. For more detailed information regarding the global page window scheme, please refer to the S12X_MMC Block Guide. |

### 7.3.3 Family ID Assignment

The family ID is a 8-bit value located in the firmware ROM (at global address: 0x7FFF0F). The read-only value is a unique family ID which is 0xC1 for S12X devices.

## 7.4 Functional Description

The BDM receives and executes commands from a host via a single wire serial interface. There are two types of BDM commands: hardware and firmware commands.

Hardware commands are used to read and write target system memory locations and to enter active background debug mode, see Section 7.4.3, “BDM Hardware Commands”. Target system memory includes all memory that is accessible by the CPU.

Firmware commands are used to read and write CPU resources and to exit from active background debug mode, see Section 7.4.4, “Standard BDM Firmware Commands”. The CPU resources referred to are the accumulator (D), X index register (X), Y index register (Y), stack pointer (SP), and program counter (PC).

Hardware commands can be executed at any time and in any mode excluding a few exceptions as highlighted (see Section 7.4.3, “BDM Hardware Commands”) and in secure mode (see Section 7.4.1, “Security”). Firmware commands can only be executed when the system is not secure and is in active background debug mode (BDM).

### 7.4.1 Security

If the user resets into special single chip mode with the system secured, a secured mode BDM firmware lookup table is brought into the map overlapping a portion of the standard BDM firmware lookup table. The secure BDM firmware verifies that the on-chip non-volatile memory (e.g. EEPROM and Flash EEPROM) is erased. This being the case, the UNSEC and ENBDM bit will get set. The BDM program jumps to the start of the standard BDM firmware and the secured mode BDM firmware is turned off and all BDM commands are allowed. If the non-volatile memory does not verify as erased, the BDM firmware sets the ENBDM bit, without asserting UNSEC, and the firmware enters a loop. This causes the BDM hardware commands to become enabled, but does not enable the firmware commands. This allows the BDM hardware to be used to erase the non-volatile memory.

BDM operation is not possible in any other mode than special single chip mode when the device is secured. The device can be unsecured via BDM serial interface in special single chip mode only. More information regarding security is provided in the security section of the device documentation.

### 7.4.2 Enabling and Activating BDM

The system must be in active BDM to execute standard BDM firmware commands. BDM can be activated only after being enabled. BDM is enabled by setting the ENBDM bit in the BDM status (BDMSTS) register. The ENBDM bit is set by writing to the BDM status (BDMSTS) register, via the single-wire interface, using a hardware command such as WRITE_BD_BYTE.

After being enabled, BDM is activated by one of the following[1]:

- Hardware BACKGROUND command
- CPU BGND instruction
- External instruction tagging mechanism[2]
- Breakpoint force or tag mechanism[2]

When BDM is activated, the CPU finishes executing the current instruction and then begins executing the firmware in the standard BDM firmware lookup table. When BDM is activated by a breakpoint, the type of breakpoint used determines if BDM becomes active before or after execution of the next instruction.

**NOTE**

If an attempt is made to activate BDM before being enabled, the CPU resumes normal instruction execution after a brief delay. If BDM is not enabled, any hardware BACKGROUND commands issued are ignored by the BDM and the CPU is not delayed.

In active BDM, the BDM registers and standard BDM firmware lookup table are mapped to addresses 0x7FFF00 to 0x7FFFFF. BDM registers are mapped to addresses 0x7FFF00 to 0x7FFF0B. The BDM uses these registers which are readable anytime by the BDM. However, these registers are not readable by user programs.

1. BDM is enabled and active immediately out of special single-chip reset.
2. This method is provided by the S12X_DBG module.

### 7.4.3 BDM Hardware Commands

Hardware commands are used to read and write target system memory locations and to enter active background debug mode. Target system memory includes all memory that is accessible by the CPU on the SOC which can be on-chip RAM, non-volatile memory (e.g. EEPROM, Flash EEPROM), I/O and control registers, and all external memory.

Hardware commands are executed with minimal or no CPU intervention and do not require the system to be in active BDM for execution, although, they can still be executed in this mode. When executing a hardware command, the BDM sub-block waits for a free bus cycle so that the background access does not disturb the running application program. If a free cycle is not found within 128 clock cycles, the CPU is momentarily frozen so that the BDM can steal a cycle. When the BDM finds a free cycle, the operation does not intrude on normal CPU operation provided that it can be completed in a single cycle. However, if an operation requires multiple cycles the CPU is frozen until the operation is complete, even though the BDM found a free cycle.

The BDM hardware commands are listed in Table 7-6.

The READ_BD and WRITE_BD commands allow access to the BDM register locations. These locations are not normally in the system memory map but share addresses with the application in memory. To distinguish between physical memory locations that share the same address, BDM memory resources are enabled just for the READ_BD and WRITE_BD access cycle. This allows the BDM to access BDM locations unobtrusively, even if the addresses conflict with the application memory map.

**Table 7-6. Hardware Commands**

| Command | Opcode (hex) | Data | Description |
|---|---|---|---|
| BACKGROUND | 90 | None | Enter background mode if firmware is enabled. If enabled, an ACK will be issued when the part enters active background mode. |
| ACK_ENABLE | D5 | None | Enable Handshake. Issues an ACK pulse after the command is executed. |
| ACK_DISABLE | D6 | None | Disable Handshake. This command does not issue an ACK pulse. |
| READ_BD_BYTE | E4 | 16-bit address<br>16-bit data out | Read from memory with standard BDM firmware lookup table in map.<br>Odd address data on low byte; even address data on high byte. |
| READ_BD_WORD | EC | 16-bit address<br>16-bit data out | Read from memory with standard BDM firmware lookup table in map.<br>Must be aligned access. |
| READ_BYTE | E0 | 16-bit address<br>16-bit data out | Read from memory with standard BDM firmware lookup table out of map.<br>Odd address data on low byte; even address data on high byte. |
| READ_WORD | E8 | 16-bit address<br>16-bit data out | Read from memory with standard BDM firmware lookup table out of map.<br>Must be aligned access. |
| WRITE_BD_BYTE | C4 | 16-bit address<br>16-bit data in | Write to memory with standard BDM firmware lookup table in map.<br>Odd address data on low byte; even address data on high byte. |
| WRITE_BD_WORD | CC | 16-bit address<br>16-bit data in | Write to memory with standard BDM firmware lookup table in map.<br>Must be aligned access. |
| WRITE_BYTE | C0 | 16-bit address<br>16-bit data in | Write to memory with standard BDM firmware lookup table out of map.<br>Odd address data on low byte; even address data on high byte. |

**Table 7-6. Hardware Commands (continued)**

| Command | Opcode (hex) | Data | Description |
|---|---|---|---|
| WRITE_WORD | C8 | 16-bit address<br>16-bit data in | Write to memory with standard BDM firmware lookup table out of map.<br>Must be aligned access. |

NOTE:
If enabled, ACK will occur when data is ready for transmission for all BDM READ commands and will occur after the write is complete for all BDM WRITE commands.

## 7.4.4 Standard BDM Firmware Commands

Firmware commands are used to access and manipulate CPU resources. The system must be in active BDM to execute standard BDM firmware commands, see Section 7.4.2, “Enabling and Activating BDM”. Normal instruction execution is suspended while the CPU executes the firmware located in the standard BDM firmware lookup table. The hardware command BACKGROUND is the usual way to activate BDM.

As the system enters active BDM, the standard BDM firmware lookup table and BDM registers become visible in the on-chip memory map at 0x7FFF00–0x7FFFFF, and the CPU begins executing the standard BDM firmware. The standard BDM firmware watches for serial commands and executes them as they are received.

The firmware commands are shown in Table 7-7.

**Table 7-7. Firmware Commands**

| Command[1] | Opcode (hex) | Data | Description |
|---|---|---|---|
| READ_NEXT[2] | 62 | 16-bit data out | Increment X index register by 2 (X = X + 2), then read word X points to. |
| READ_PC | 63 | 16-bit data out | Read program counter. |
| READ_D | 64 | 16-bit data out | Read D accumulator. |
| READ_X | 65 | 16-bit data out | Read X index register. |
| READ_Y | 66 | 16-bit data out | Read Y index register. |
| READ_SP | 67 | 16-bit data out | Read stack pointer. |
| WRITE_NEXT | 42 | 16-bit data in | Increment X index register by 2 (X = X + 2), then write word to location pointed to by X. |
| WRITE_PC | 43 | 16-bit data in | Write program counter. |
| WRITE_D | 44 | 16-bit data in | Write D accumulator. |
| WRITE_X | 45 | 16-bit data in | Write X index register. |
| WRITE_Y | 46 | 16-bit data in | Write Y index register. |
| WRITE_SP | 47 | 16-bit data in | Write stack pointer. |
| GO | 08 | none | Go to user program. If enabled, ACK will occur when leaving active background mode. |
| GO_UNTIL[3] | 0C | none | Go to user program. If enabled, ACK will occur upon returning to active background mode. |
| TRACE1 | 10 | none | Execute one user instruction then return to active BDM. If enabled, ACK will occur upon returning to active background mode. |
| TAGGO -> GO | 18 | none | (Previous enable tagging and go to user program.)<br>This command will be deprecated and should not be used anymore.<br>Opcode will be executed as a GO command. |

1. If enabled, ACK will occur when data is ready for transmission for all BDM READ commands and will occur after the write is complete for all BDM WRITE commands.
2. When the firmware command READ_NEXT or WRITE_NEXT is used to access the BDM address space the BDM resources are accessed rather than user code. Writing BDM firmware is not possible.
3. System stop disables the ACK function and ignored commands will not have an ACK-pulse (e.g., CPU in stop or wait mode). The GO_UNTIL command will not get an Acknowledge if CPU executes the wait or stop instruction before the “UNTIL” condition (BDM active again) is reached (see Section 7.4.7, “Serial Interface Hardware Handshake Protocol” last Note).

## 7.4.5 BDM Command Structure

Hardware and firmware BDM commands start with an 8-bit opcode followed by a 16-bit address and/or a 16-bit data word depending on the command. All the read commands return 16 bits of data despite the byte or word implication in the command name.

8-bit reads return 16-bits of data, of which, only one byte will contain valid data. If reading an even address, the valid data will appear in the MSB. If reading an odd address, the valid data will appear in the LSB.

16-bit misaligned reads and writes are generally not allowed. If attempted by BDM hardware command, the BDM will ignore the least significant bit of the address and will assume an even address from the remaining bits.

For devices with external bus:

The following cycle count information is only valid when the external wait function is not used (see wait bit of EBI sub-block). During an external wait the BDM can not steal a cycle. Hence be careful with the external wait function if the BDM serial interface is much faster than the bus, because of the BDM soft-reset after time-out (see Section 7.4.11, “Serial Communication Time Out”).

For hardware data read commands, the external host must wait at least 150 bus clock cycles after sending the address before attempting to obtain the read data. This is to be certain that valid data is available in the BDM shift register, ready to be shifted out. For hardware write commands, the external host must wait 150 bus clock cycles after sending the data to be written before attempting to send a new command. This is to avoid disturbing the BDM shift register before the write has been completed. The 150 bus clock cycle delay in both cases includes the maximum 128 cycle delay that can be incurred as the BDM waits for a free cycle before stealing a cycle.

For firmware read commands, the external host should wait at least 48 bus clock cycles after sending the command opcode and before attempting to obtain the read data. This includes the potential of extra cycles when the access is external and stretched (+1 to maximum +7 cycles) or to registers of the PRU (port replacement unit) in emulation modes (if modes available). The 48 cycle wait allows enough time for the requested data to be made available in the BDM shift register, ready to be shifted out.

**NOTE**

This timing has increased from previous BDM modules due to the new capability in which the BDM serial interface can potentially run faster than the bus. On previous BDM modules this extra time could be hidden within the serial time.

For firmware write commands, the external host must wait 36 bus clock cycles after sending the data to be written before attempting to send a new command. This is to avoid disturbing the BDM shift register before the write has been completed.

The external host should wait at least for 76 bus clock cycles after a TRACE1 or GO command before starting any new serial command. This is to allow the CPU to exit gracefully from the standard BDM firmware lookup table and resume execution of the user code. Disturbing the BDM shift register prematurely may adversely affect the exit from the standard BDM firmware lookup table.

**NOTE**

If the bus rate of the target processor is unknown or could be changing or the external wait function is used, it is recommended that the ACK (acknowledge function) is used to indicate when an operation is complete. When using ACK, the delay times are automated.

Figure 7-7 represents the BDM command structure. The command blocks illustrate a series of eight bit times starting with a falling edge. The bar across the top of the blocks indicates that the BKGD line idles in the high state. The time for an 8-bit command is $8 \times 16$ target clock cycles.[1]

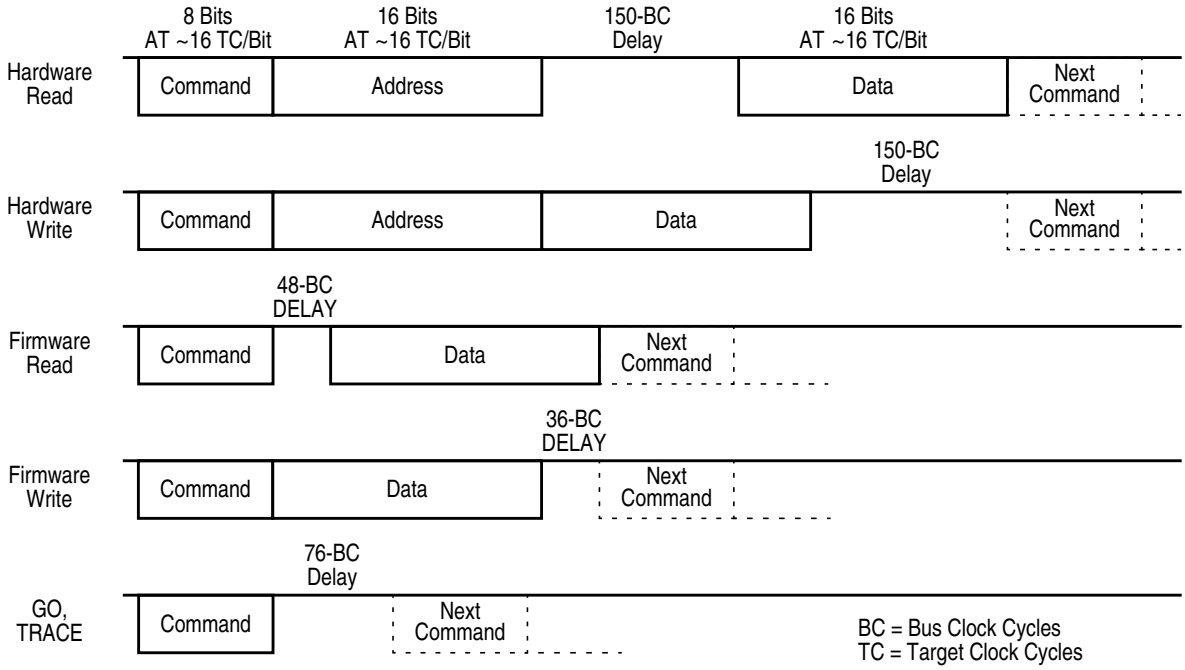



**Figure 7-7. BDM Command Structure**

## 7.4.6 BDM Serial Interface

The BDM communicates with external devices serially via the BKGD pin. During reset, this pin is a mode select input which selects between normal and special modes of operation. After reset, this pin becomes the dedicated serial interface pin for the BDM.

The BDM serial interface is timed using the clock selected by the CLKSW bit in the status register see Section 7.3.2.1, “BDM Status Register (BDMSTS)”. This clock will be referred to as the target clock in the following explanation.

The BDM serial interface uses a clocking scheme in which the external host generates a falling edge on the BKGD pin to indicate the start of each bit time. This falling edge is sent for every bit whether data is transmitted or received. Data is transferred most significant bit (MSB) first at 16 target clock cycles per bit. The interface times out if 512 clock cycles occur between falling edges from the host.

The BKGD pin is a pseudo open-drain pin and has an weak on-chip active pull-up that is enabled at all times. It is assumed that there is an external pull-up and that drivers connected to BKGD do not typically drive the high level. Since R-C rise time could be unacceptably long, the target system and host provide brief driven-high (speedup) pulses to drive BKGD to a logic 1. The source of this speedup pulse is the host for transmit cases and the target for receive cases.

The timing for host-to-target is shown in Figure 7-8 and that of target-to-host in Figure 7-9 and Figure 7-10. All four cases begin when the host drives the BKGD pin low to generate a falling edge. Since the host and target are operating from separate clocks, it can take the target system up to one full clock

1. Target clock cycles are cycles measured using the target MCU’s serial clock rate. See Section 7.4.6, “BDM Serial Interface” and Section 7.3.2.1, “BDM Status Register (BDMSTS)” for information on how serial clock rate is selected.

cycle to recognize this edge. The target measures delays from this perceived start of the bit time while the host measures delays from the point it actually drove BKGD low to start the bit up to one target clock cycle earlier. Synchronization between the host and target is established in this manner at the start of every bit time.

Figure 7-8 shows an external host transmitting a logic 1 and transmitting a logic 0 to the BKGD pin of a target system. The host is asynchronous to the target, so there is up to a one clock-cycle delay from the host-generated falling edge to where the target recognizes this edge as the beginning of the bit time. Ten target clock cycles later, the target senses the bit level on the BKGD pin. Internal glitch detect logic requires the pin be driven high no later that eight target clock cycles after the falling edge for a logic 1 transmission.

Since the host drives the high speedup pulses in these two cases, the rising edges look like digitally driven signals.

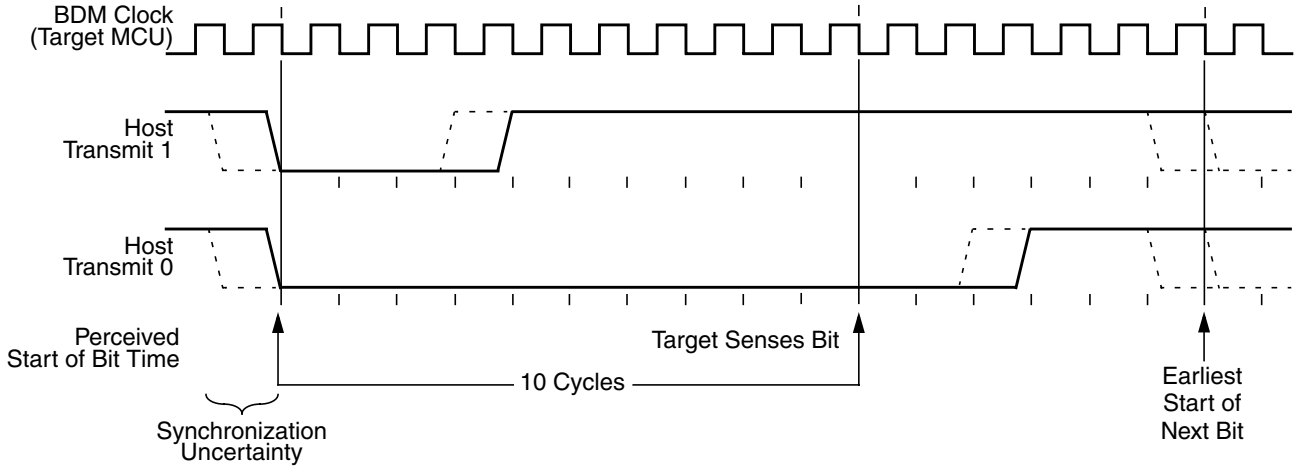



**Figure 7-8. BDM Host-to-Target Serial Bit Timing**

The receive cases are more complicated. Figure 7-9 shows the host receiving a logic 1 from the target system. Since the host is asynchronous to the target, there is up to one clock-cycle delay from the host-generated falling edge on BKGD to the perceived start of the bit time in the target. The host holds the BKGD pin low long enough for the target to recognize it (at least two target clock cycles). The host must release the low drive before the target drives a brief high speedup pulse seven target clock cycles after the perceived start of the bit time. The host should sample the bit level about 10 target clock cycles after it started the bit time.

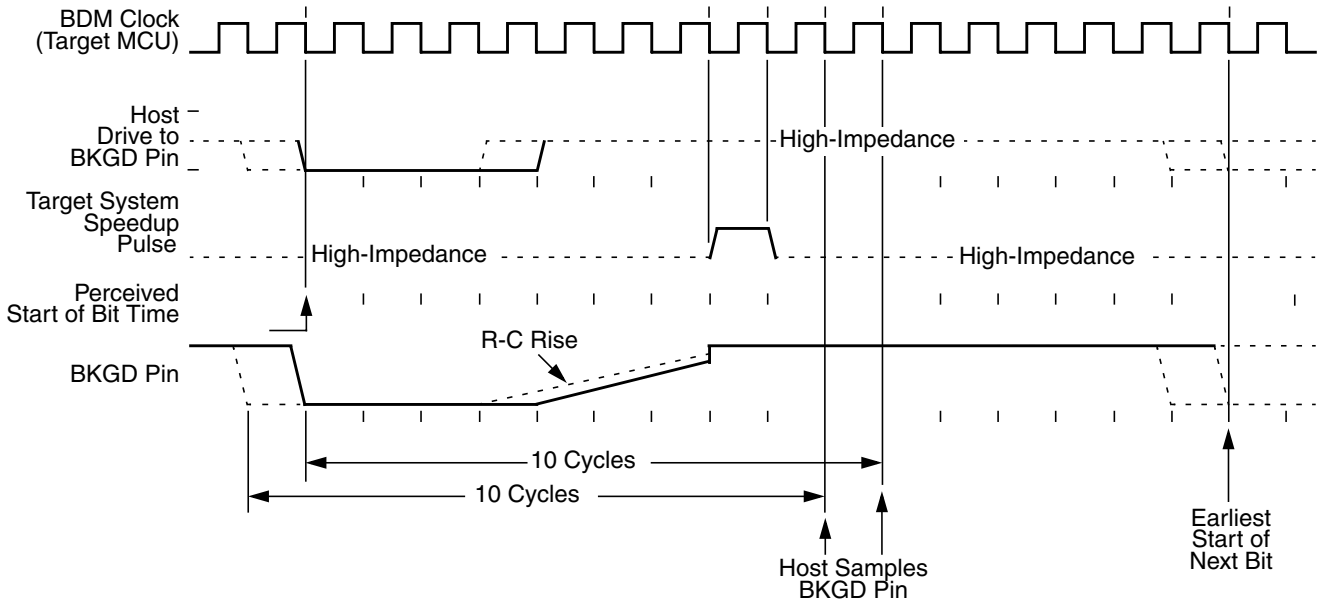



**Figure 7-9. BDM Target-to-Host Serial Bit Timing (Logic 1)**

Figure 7-10 shows the host receiving a logic 0 from the target. Since the host is asynchronous to the target, there is up to a one clock-cycle delay from the host-generated falling edge on BKGD to the start of the bit time as perceived by the target. The host initiates the bit time but the target finishes it. Since the target wants the host to receive a logic 0, it drives the BKGD pin low for 13 target clock cycles then briefly drives it high to speed up the rising edge. The host samples the bit level about 10 target clock cycles after starting the bit time.

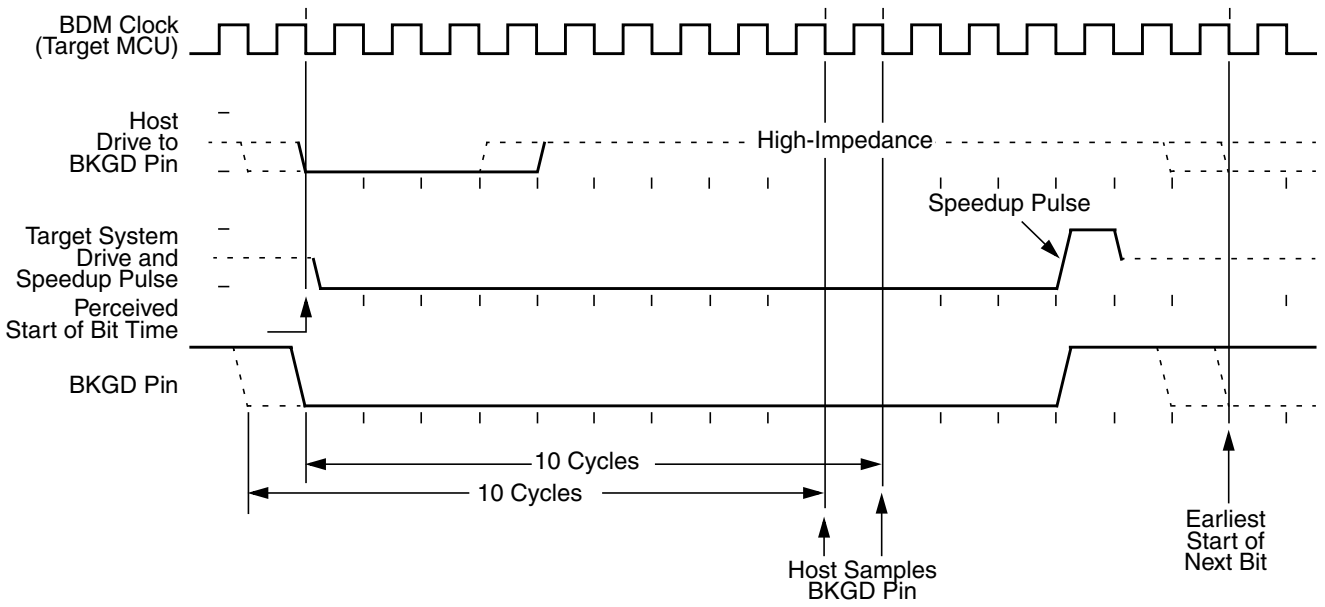



**Figure 7-10. BDM Target-to-Host Serial Bit Timing (Logic 0)**

## 7.4.7 Serial Interface Hardware Handshake Protocol

BDM commands that require CPU execution are ultimately treated at the MCU bus rate. Since the BDM clock source can be asynchronously related to the bus frequency, when CLKSW = 0, it is very helpful to provide a handshake protocol in which the host could determine when an issued command is executed by the CPU. The alternative is to always wait the amount of time equal to the appropriate number of cycles at the slowest possible rate the clock could be running. This sub-section will describe the hardware handshake protocol.

The hardware handshake protocol signals to the host controller when an issued command was successfully executed by the target. This protocol is implemented by a 16 serial clock cycle low pulse followed by a brief speedup pulse in the BKGD pin. This pulse is generated by the target MCU when a command, issued by the host, has been successfully executed (see Figure 7-11). This pulse is referred to as the ACK pulse. After the ACK pulse has finished: the host can start the bit retrieval if the last issued command was a read command, or start a new command if the last command was a write command or a control command (BACKGROUND, GO, GO_UNTIL or TRACE1). The ACK pulse is not issued earlier than 32 serial clock cycles after the BDM command was issued. The end of the BDM command is assumed to be the 16th tick of the last bit. This minimum delay assures enough time for the host to perceive the ACK pulse. Note also that, there is no upper limit for the delay between the command and the related ACK pulse, since the command execution depends upon the CPU bus frequency, which in some cases could be very slow

compared to the serial communication rate. This protocol allows a great flexibility for the POD designers, since it does not rely on any accurate time measurement or short response time to any event in the serial communication.

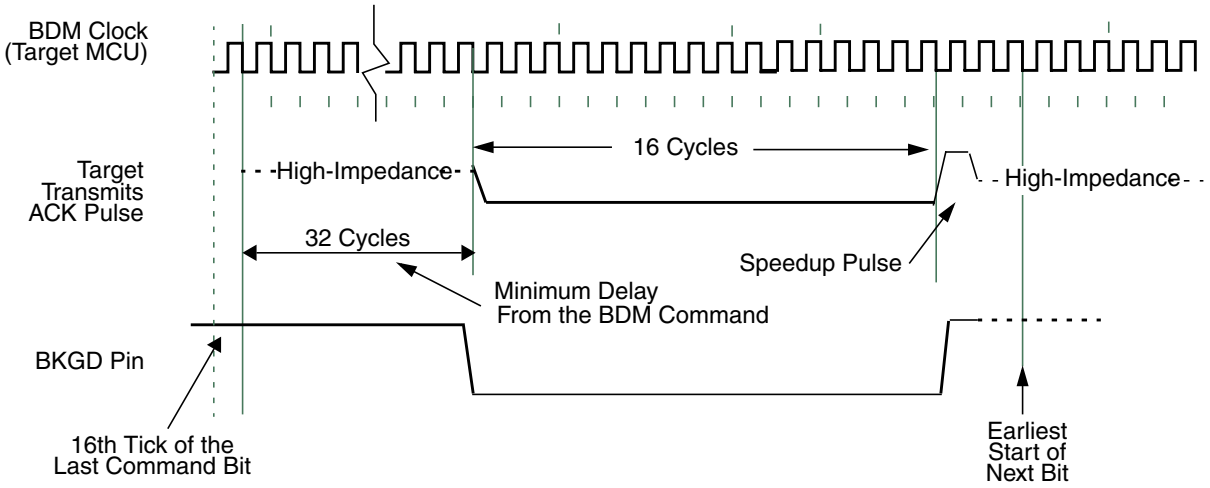



**Figure 7-11. Target Acknowledge Pulse (ACK)**

**NOTE**

If the ACK pulse was issued by the target, the host assumes the previous command was executed. If the CPU enters wait or stop prior to executing a hardware command, the ACK pulse will not be issued meaning that the BDM command was not executed. After entering wait or stop mode, the BDM command is no longer pending.

Figure 7-12 shows the ACK handshake protocol in a command level timing diagram. The READ_BYTE instruction is used as an example. First, the 8-bit instruction opcode is sent by the host, followed by the address of the memory location to be read. The target BDM decodes the instruction. A bus cycle is grabbed (free or stolen) by the BDM and it executes the READ_BYTE operation. Having retrieved the data, the BDM issues an ACK pulse to the host controller, indicating that the addressed byte is ready to be retrieved. After detecting the ACK pulse, the host initiates the byte retrieval process. Note that data is sent in the form of a word and the host needs to determine which is the appropriate byte based on whether the address was odd or even.

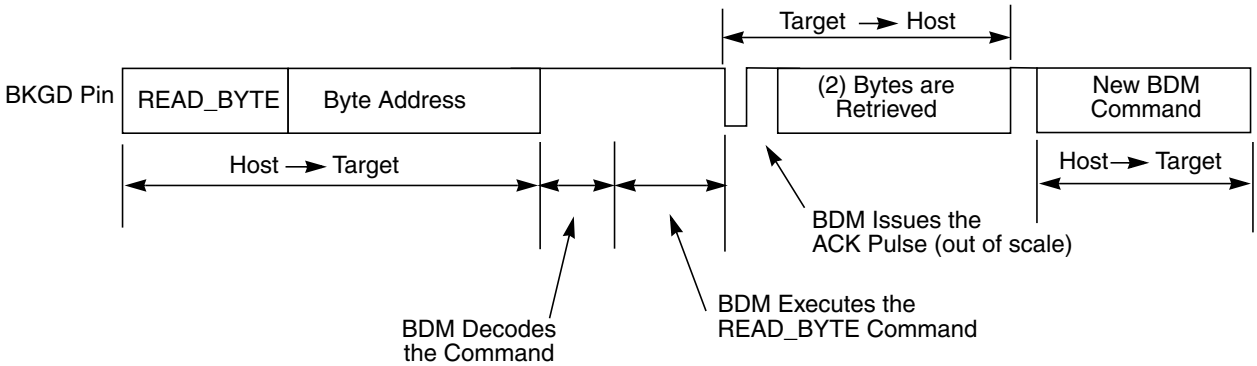



**Figure 7-12. Handshake Protocol at Command Level**

Differently from the normal bit transfer (where the host initiates the transmission), the serial interface ACK handshake pulse is initiated by the target MCU by issuing a negative edge in the BKGD pin. The hardware handshake protocol in Figure 7-11 specifies the timing when the BKGD pin is being driven, so the host should follow this timing constraint in order to avoid the risk of an electrical conflict in the BKGD pin.

**NOTE**

The only place the BKGD pin can have an electrical conflict is when one side is driving low and the other side is issuing a speedup pulse (high). Other "highs" are pulled rather than driven. However, at low rates the time of the speedup pulse can become lengthy and so the potential conflict time becomes longer as well.

The ACK handshake protocol does not support nested ACK pulses. If a BDM command is not acknowledge by an ACK pulse, the host needs to abort the pending command first in order to be able to issue a new BDM command. When the CPU enters wait or stop while the host issues a hardware command (e.g., WRITE_BYTE), the target discards the incoming command due to the wait or stop being detected. Therefore, the command is not acknowledged by the target, which means that the ACK pulse will not be issued in this case. After a certain time the host (not aware of stop or wait) should decide to abort any possible pending ACK pulse in order to be sure a new command can be issued. Therefore, the protocol provides a mechanism in which a command, and its corresponding ACK, can be aborted.

**NOTE**

The ACK pulse does not provide a time out. This means for the GO_UNTIL command that it can not be distinguished if a stop or wait has been executed (command discarded and ACK not issued) or if the "UNTIL" condition (BDM active) is just not reached yet. Hence in any case where the ACK pulse of a command is not issued the possible pending command should be aborted before issuing a new command. See the handshake abort procedure described in Section 7.4.8, "Hardware Handshake Abort Procedure".

### 7.4.8 Hardware Handshake Abort Procedure

The abort procedure is based on the SYNC command. In order to abort a command, which had not issued the corresponding ACK pulse, the host controller should generate a low pulse in the BKGD pin by driving it low for at least 128 serial clock cycles and then driving it high for one serial clock cycle, providing a speedup pulse. By detecting this long low pulse in the BKGD pin, the target executes the SYNC protocol, see Section 7.4.9, "SYNC — Request Timed Reference Pulse", and assumes that the pending command and therefore the related ACK pulse, are being aborted. Therefore, after the SYNC protocol has been completed the host is free to issue new BDM commands. For Firmware READ or WRITE commands it can not be guaranteed that the pending command is aborted when issuing a SYNC before the corresponding ACK pulse. There is a short latency time from the time the READ or WRITE access begins until it is finished and the corresponding ACK pulse is issued. The latency time depends on the firmware READ or WRITE command that is issued and if the serial interface is running on a different clock rate than the bus. When the SYNC command starts during this latency time the READ or WRITE command will not be aborted, but the corresponding ACK pulse will be aborted. A pending GO, TRACE1 or

GO_UNTIL command can not be aborted. Only the corresponding ACK pulse can be aborted by the SYNC command.

Although it is not recommended, the host could abort a pending BDM command by issuing a low pulse in the BKGD pin shorter than 128 serial clock cycles, which will not be interpreted as the SYNC command. The ACK is actually aborted when a negative edge is perceived by the target in the BKGD pin. The short abort pulse should have at least 4 clock cycles keeping the BKGD pin low, in order to allow the negative edge to be detected by the target. In this case, the target will not execute the SYNC protocol but the pending command will be aborted along with the ACK pulse. The potential problem with this abort procedure is when there is a conflict between the ACK pulse and the short abort pulse. In this case, the target may not perceive the abort pulse. The worst case is when the pending command is a read command (i.e., READ_BYTE). If the abort pulse is not perceived by the target the host will attempt to send a new command after the abort pulse was issued, while the target expects the host to retrieve the accessed memory byte. In this case, host and target will run out of synchronism. However, if the command to be aborted is not a read command the short abort pulse could be used. After a command is aborted the target assumes the next negative edge, after the abort pulse, is the first bit of a new BDM command.

**NOTE**

The details about the short abort pulse are being provided only as a reference for the reader to better understand the BDM internal behavior. It is not recommended that this procedure be used in a real application.

Since the host knows the target serial clock frequency, the SYNC command (used to abort a command) does not need to consider the lower possible target frequency. In this case, the host could issue a SYNC very close to the 128 serial clock cycles length. Providing a small overhead on the pulse length in order to assure the SYNC pulse will not be misinterpreted by the target. See Section 7.4.9, “SYNC — Request Timed Reference Pulse”.

Figure 7-13 shows a SYNC command being issued after a READ_BYTE, which aborts the READ_BYTE command. Note that, after the command is aborted a new command could be issued by the host computer.

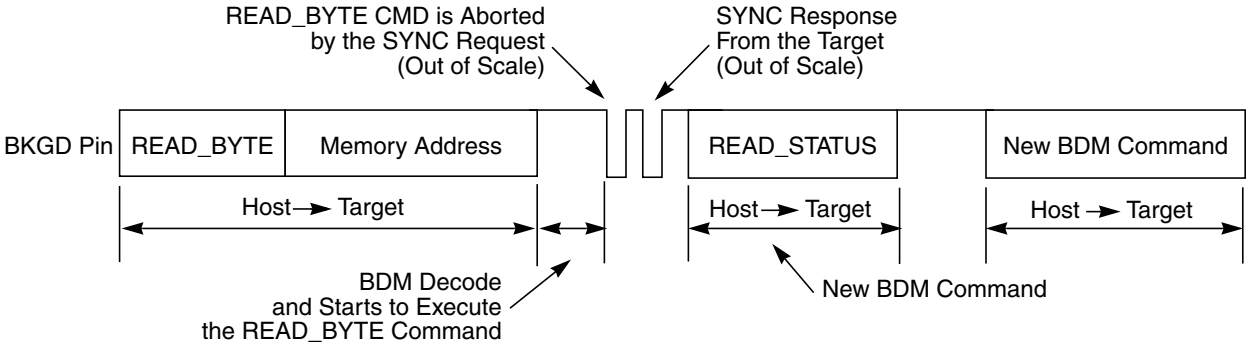



**Figure 7-13. ACK Abort Procedure at the Command Level**

**NOTE**

Figure 7-13 does not represent the signals in a true timing scale

Figure 7-14 shows a conflict between the ACK pulse and the SYNC request pulse. This conflict could occur if a POD device is connected to the target BKGD pin and the target is already in debug active mode.

Consider that the target CPU is executing a pending BDM command at the exact moment the POD is being connected to the BKGD pin. In this case, an ACK pulse is issued along with the SYNC command. In this case, there is an electrical conflict between the ACK speedup pulse and the SYNC pulse. Since this is not a probable situation, the protocol does not prevent this conflict from happening.

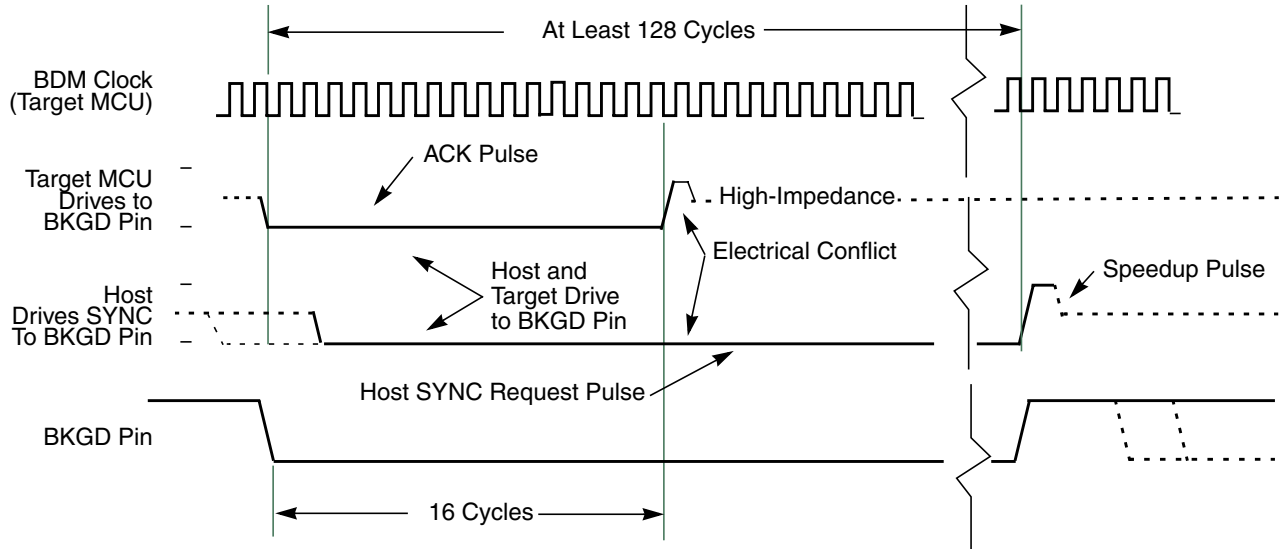



**Figure 7-14. ACK Pulse and SYNC Request Conflict**

**NOTE**

This information is being provided so that the MCU integrator will be aware that such a conflict could eventually occur.

The hardware handshake protocol is enabled by the ACK_ENABLE and disabled by the ACK_DISABLE BDM commands. This provides backwards compatibility with the existing POD devices which are not able to execute the hardware handshake protocol. It also allows for new POD devices, that support the hardware handshake protocol, to freely communicate with the target device. If desired, without the need for waiting for the ACK pulse.

The commands are described as follows:

- ACK_ENABLE — enables the hardware handshake protocol. The target will issue the ACK pulse when a CPU command is executed by the CPU. The ACK_ENABLE command itself also has the ACK pulse as a response.
- ACK_DISABLE — disables the ACK pulse protocol. In this case, the host needs to use the worst case delay time at the appropriate places in the protocol.

The default state of the BDM after reset is hardware handshake protocol disabled.

All the read commands will ACK (if enabled) when the data bus cycle has completed and the data is then ready for reading out by the BKGD serial pin. All the write commands will ACK (if enabled) after the data has been received by the BDM through the BKGD serial pin and when the data bus cycle is complete. See Section 7.4.3, “BDM Hardware Commands” and Section 7.4.4, “Standard BDM Firmware Commands” for more information on the BDM commands.

The ACK_ENABLE sends an ACK pulse when the command has been completed. This feature could be used by the host to evaluate if the target supports the hardware handshake protocol. If an ACK pulse is issued in response to this command, the host knows that the target supports the hardware handshake protocol. If the target does not support the hardware handshake protocol the ACK pulse is not issued. In this case, the ACK_ENABLE command is ignored by the target since it is not recognized as a valid command.

The BACKGROUND command will issue an ACK pulse when the CPU changes from normal to background mode. The ACK pulse related to this command could be aborted using the SYNC command.

The GO command will issue an ACK pulse when the CPU exits from background mode. The ACK pulse related to this command could be aborted using the SYNC command.

The GO_UNTIL command is equivalent to a GO command with exception that the ACK pulse, in this case, is issued when the CPU enters into background mode. This command is an alternative to the GO command and should be used when the host wants to trace if a breakpoint match occurs and causes the CPU to enter active background mode. Note that the ACK is issued whenever the CPU enters BDM, which could be caused by a breakpoint match or by a BGND instruction being executed. The ACK pulse related to this command could be aborted using the SYNC command.

The TRACE1 command has the related ACK pulse issued when the CPU enters background active mode after one instruction of the application program is executed. The ACK pulse related to this command could be aborted using the SYNC command.

### 7.4.9 SYNC — Request Timed Reference Pulse

The SYNC command is unlike other BDM commands because the host does not necessarily know the correct communication speed to use for BDM communications until after it has analyzed the response to the SYNC command. To issue a SYNC command, the host should perform the following steps:

1. Drive the BKGD pin low for at least 128 cycles at the lowest possible BDM serial communication frequency (the lowest serial communication frequency is determined by the crystal oscillator or the clock chosen by CLKSW.)
2. Drive BKGD high for a brief speedup pulse to get a fast rise time (this speedup pulse is typically one cycle of the host clock.)
3. Remove all drive to the BKGD pin so it reverts to high impedance.
4. Listen to the BKGD pin for the sync response pulse.

Upon detecting the SYNC request from the host, the target performs the following steps:

1. Discards any incomplete command received or bit retrieved.
2. Waits for BKGD to return to a logic one.
3. Delays 16 cycles to allow the host to stop driving the high speedup pulse.
4. Drives BKGD low for 128 cycles at the current BDM serial communication frequency.
5. Drives a one-cycle high speedup pulse to force a fast rise time on BKGD.
6. Removes all drive to the BKGD pin so it reverts to high impedance.

The host measures the low time of this 128 cycle SYNC response pulse and determines the correct speed for subsequent BDM communications. Typically, the host can determine the correct communication speed

within a few percent of the actual target speed and the communication protocol can easily tolerate speed errors of several percent.

As soon as the SYNC request is detected by the target, any partially received command or bit retrieved is discarded. This is referred to as a soft-reset, equivalent to a time-out in the serial communication. After the SYNC response, the target will consider the next negative edge (issued by the host) as the start of a new BDM command or the start of new SYNC request.

Another use of the SYNC command pulse is to abort a pending ACK pulse. The behavior is exactly the same as in a regular SYNC command. Note that one of the possible causes for a command to not be acknowledged by the target is a host-target synchronization problem. In this case, the command may not have been understood by the target and so an ACK response pulse will not be issued.

### 7.4.10 Instruction Tracing

When a TRACE1 command is issued to the BDM in active BDM, the CPU exits the standard BDM firmware and executes a single instruction in the user code. Once this has occurred, the CPU is forced to return to the standard BDM firmware and the BDM is active and ready to receive a new command. If the TRACE1 command is issued again, the next user instruction will be executed. This facilitates stepping or tracing through the user code one instruction at a time.

If an interrupt is pending when a TRACE1 command is issued, the interrupt stacking operation occurs but no user instruction is executed. Once back in standard BDM firmware execution, the program counter points to the first instruction in the interrupt service routine.

Be aware when tracing through the user code that the execution of the user code is done step by step but all peripherals are free running. Hence possible timing relations between CPU code execution and occurrence of events of other peripherals no longer exist.

Do not trace the CPU instruction BGND used for soft breakpoints. Tracing the BGND instruction will result in a return address pointing to BDM firmware address space.

When tracing through user code which contains stop or wait instructions the following will happen when the stop or wait instruction is traced:

The CPU enters stop or wait mode and the TRACE1 command can not be finished before leaving the low power mode. This is the case because BDM active mode can not be entered after CPU executed the stop instruction. However all BDM hardware commands except the BACKGROUND command are operational after tracing a stop or wait instruction and still being in stop or wait mode. If system stop mode is entered (all bus masters are in stop mode) no BDM command is operational.

As soon as stop or wait mode is exited the CPU enters BDM active mode and the saved PC value points to the entry of the corresponding interrupt service routine.

In case the handshake feature is enabled the corresponding ACK pulse of the TRACE1 command will be discarded when tracing a stop or wait instruction. Hence there is no ACK pulse when BDM active mode is entered as part of the TRACE1 command after CPU exited from stop or wait mode. All valid commands sent during CPU being in stop or wait mode or after CPU exited from stop or wait mode will have an ACK pulse. The handshake feature becomes disabled only when system stop mode has been reached. Hence

after a system stop mode the handshake feature must be enabled again by sending the ACK_ENABLE command.

### 7.4.11 Serial Communication Time Out

The host initiates a host-to-target serial transmission by generating a falling edge on the BKGD pin. If BKGD is kept low for more than 128 target clock cycles, the target understands that a SYNC command was issued. In this case, the target will keep waiting for a rising edge on BKGD in order to answer the SYNC request pulse. If the rising edge is not detected, the target will keep waiting forever without any time-out limit.

Consider now the case where the host returns BKGD to logic one before 128 cycles. This is interpreted as a valid bit transmission, and not as a SYNC request. The target will keep waiting for another falling edge marking the start of a new bit. If, however, a new falling edge is not detected by the target within 512 clock cycles since the last falling edge, a time-out occurs and the current command is discarded without affecting memory or the operating mode of the MCU. This is referred to as a soft-reset.

If a read command is issued but the data is not retrieved within 512 serial clock cycles, a soft-reset will occur causing the command to be disregarded. The data is not available for retrieval after the time-out has occurred. This is the expected behavior if the handshake protocol is not enabled. However, consider the behavior where the BDM is running in a frequency much greater than the CPU frequency. In this case, the command could time out before the data is ready to be retrieved. In order to allow the data to be retrieved even with a large clock frequency mismatch (between BDM and CPU) when the hardware handshake protocol is enabled, the time out between a read command and the data retrieval is disabled. Therefore, the host could wait for more then 512 serial clock cycles and still be able to retrieve the data from an issued read command. However, once the handshake pulse (ACK pulse) is issued, the time-out feature is re-activated, meaning that the target will time out after 512 clock cycles. Therefore, the host needs to retrieve the data within a 512 serial clock cycles time frame after the ACK pulse had been issued. After that period, the read command is discarded and the data is no longer available for retrieval. Any negative edge in the BKGD pin after the time-out period is considered to be a new command or a SYNC request.

Note that whenever a partially issued command, or partially retrieved data, has occurred the time out in the serial communication is active. This means that if a time frame higher than 512 serial clock cycles is observed between two consecutive negative edges and the command being issued or data being retrieved is not complete, a soft-reset will occur causing the partially received command or data retrieved to be disregarded. The next negative edge in the BKGD pin, after a soft-reset has occurred, is considered by the target as the start of a new BDM command, or the start of a SYNC request pulse.

# Chapter 8
# S12X Debug (S12XDBGV3) Module

**Table 8-1. Revision History**

| Revision Number | Revision Date | Sections Affected | Description of Changes |
|---|---|---|---|
| V03.20 | 14 Sep 2007 | 8.3.2.7/8-317 | - Clarified reserved State Sequencer encodings. |
| V03.21 | 23 Oct 2007 | 8.4.2.2/8-329<br>8.4.2.4/8-330 | - Added single databyte comparison limitation information<br>- Added statement about interrupt vector fetches whilst tagging. |
| V03.22 | 12 Nov 2007 | 8.4.5.2/8-334<br>8.4.5.5/8-341 | - Removed LOOP1 tracing restriction NOTE.<br>- Added pin reset effect NOTE. |
| V03.23 | 13 Nov 2007 | General | - Text readability improved, typo removed. |
| V03.24 | 04 Jan 2008 | 8.4.5.3/8-336 | - Corrected bit name. |
| V03.25 | 14 May 2008 | General | - Updated Revision History Table format. Corrected other paragraph formats. |
| V03.26 | 12 Sep 2012 | General | - Added missing full stops. Removed redundant quotation marks. |

## 8.1 Introduction

The S12XDBG module provides an on-chip trace buffer with flexible triggering capability to allow non-intrusive debug of application software. The S12XDBG module is optimized for the S12X 16-bit architecture and allows debugging of CPU12Xand XGATE module operations.

Typically the S12XDBG module is used in conjunction with the S12XBDM module, whereby the user configures the S12XDBG module for a debugging session over the BDM interface. Once configured the S12XDBG module is armed and the device leaves BDM Mode returning control to the user program, which is then monitored by the S12XDBG module. Alternatively the S12XDBG module can be configured over a serial interface using SWI routines.

### 8.1.1 Glossary

**Table 8-2. Glossary Of Terms**

| Term | Definition |
|---|---|
| COF | Change Of Flow.<br>Change in the program flow due to a conditional branch, indexed jump or interrupt |
| BDM | Background Debug Mode |
| DUG | Device User Guide, describing the features of the device into which the DBG is integrated |
| WORD | 16-bit data entity |

**Table 8-2. Glossary Of Terms (continued)**

| Term | Definition |
|---|---|
| Data Line | 64-bit data entity |
| CPU | CPU12X module |
| Tag | Tags can be attached to XGATE or CPU opcodes as they enter the instruction pipe. If the tagged opcode reaches the execution stage a tag hit occurs. |

## 8.1.2 Overview

The comparators monitor the bus activity of the CPU12X and XGATE. When a match occurs the control logic can trigger the state sequencer to a new state. On a transition to the Final State, bus tracing is triggered and/or a breakpoint can be generated.

Independent of comparator matches a transition to Final State with associated tracing and breakpoint can be triggered by the external $\overline{\text{TAGHI}}$ and $\overline{\text{TAGLO}}$ signals, or by an XGATE module S/W breakpoint request or by writing to the TRIG control bit.

The trace buffer is visible through a 2-byte window in the register address map and can be read out using standard 16-bit word reads. Tracing is disabled when the MCU system is secured.

## 8.1.3 Features

- Four comparators (A, B, C, and D)
  - Comparators A and C compare the full address bus and full 16-bit data bus
  - Comparators A and C feature a data bus mask register
  - Comparators B and D compare the full address bus only
  - Each comparator can be configured to monitor CPU12X or XGATE buses
  - Each comparator features selection of read or write access cycles
  - Comparators B and D allow selection of byte or word access cycles
  - Comparisons can be used as triggers for the state sequencer
- Three comparator modes
  - Simple address/data comparator match mode
  - Inside address range mode, Addmin ≤ Address ≤ Addmax
  - Outside address range match mode, Address < Addmin or Address > Addmax
- Two types of triggers
  - Tagged — This triggers just before a specific instruction begins execution
  - Force — This triggers on the first instruction boundary after a match occurs.
- The following types of breakpoints
  - CPU12X breakpoint entering BDM on breakpoint (BDM)
  - CPU12X breakpoint executing SWI on breakpoint (SWI)
  - XGATE breakpoint
- External CPU12X instruction tagging trigger independent of comparators

- XGATE S/W breakpoint request trigger independent of comparators
- TRIG Immediate software trigger independent of comparators
- Four trace modes
  - Normal: change of flow (COF) PC information is stored (see Section 8.4.5.2.1) for change of flow definition.
  - Loop1: same as Normal but inhibits consecutive duplicate source address entries
  - Detail: address and data for all cycles except free cycles and opcode fetches are stored
  - Pure PC: All program counter addresses are stored.
- 4-stage state sequencer for trace buffer control
  - Tracing session trigger linked to Final State of state sequencer
  - Begin, End, and Mid alignment of tracing to trigger

## 8.1.4 Modes of Operation

The S12XDBG module can be used in all MCU functional modes.

During BDM hardware accesses and whilst the BDM module is active, CPU12X monitoring is disabled. Thus breakpoints, comparators, and CPU12X bus tracing are disabled but XGATE bus monitoring accessing the S12XDBG registers, including comparator registers, is still possible. While in active BDM or during hardware BDM accesses, XGATE activity can still be compared, traced and can be used to generate a breakpoint to the XGATE module. When the CPU12X enters active BDM Mode through a BACKGROUND command, with the S12XDBG module armed, the S12XDBG remains armed.

The S12XDBG module tracing is disabled if the MCU is secure. However, breakpoints can still be generated if the MCU is secure.

**Table 8-3. Mode Dependent Restriction Summary**

| BDM Enable | BDM Active | MCU Secure | Comparator Matches Enabled | Breakpoints Possible | Tagging Possible | Tracing Possible |
|---|---|---|---|---|---|---|
| x | x | 1 | Yes | Yes | Yes | No |
| 0 | 0 | 0 | Yes | Only SWI | Yes | Yes |
| 0 | 1 | 0 | Active BDM not possible when not enabled | | | |
| 1 | 0 | 0 | Yes | Yes | Yes | Yes |
| 1 | 1 | 0 | XGATE only | XGATE only | XGATE only | XGATE only |

### 8.1.5 Block Diagram

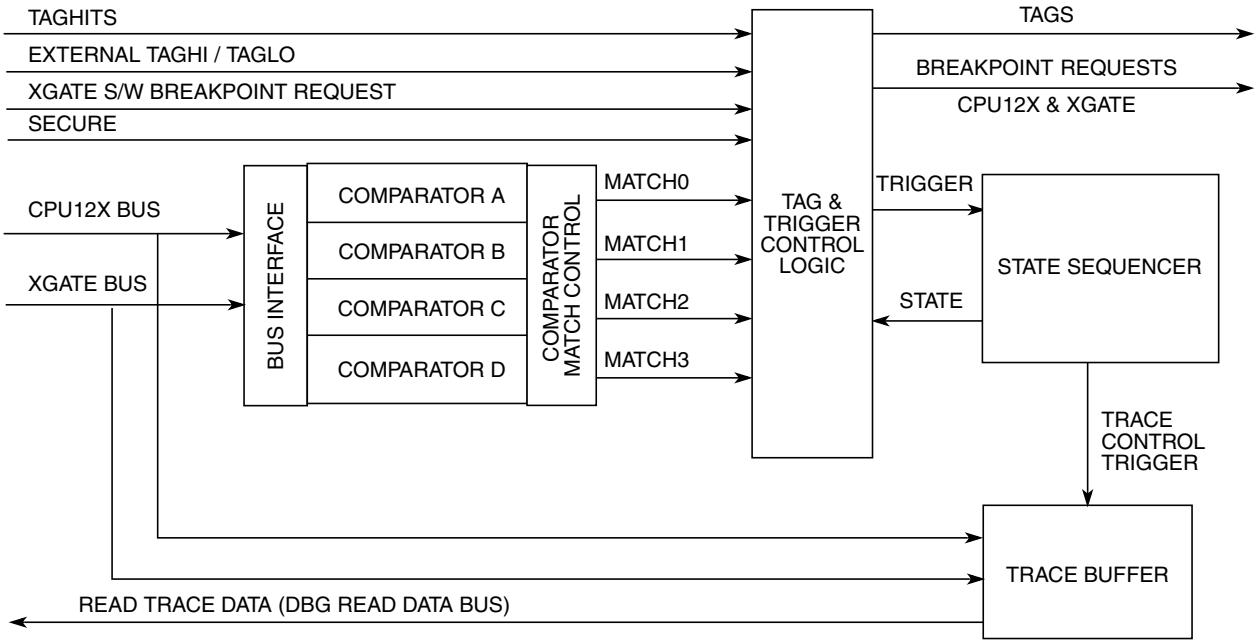



**Figure 8-1. Debug Module Block Diagram**

## 8.2 External Signal Description

The S12XDBG sub-module features two external tag input signals. See Device User Guide (DUG) for the mapping of these signals to device pins. These tag pins may be used for the external tagging in emulation modes only.

**Table 8-4. External System Pins Associated With S12XDBG**

| Pin Name | Pin Functions | Description |
|---|---|---|
| $\overline{\text{TAGHI}}$ (See DUG) | TAGHI | When instruction tagging is on, tags the high half of the instruction word being read into the instruction queue. |
| $\overline{\text{TAGLO}}$ (See DUG) | TAGLO | When instruction tagging is on, tags the low half of the instruction word being read into the instruction queue. |
| $\overline{\text{TAGLO}}$ (See DUG) | Unconditional Tagging Enable | In emulation modes, a low assertion on this pin in the 7th or 8th cycle after the end of reset enables the Unconditional Tagging function. |

## 8.3 Memory Map and Registers

### 8.3.1 Module Memory Map

A summary of the registers associated with the S12XDBG sub-block is shown in Table 8-2. Detailed descriptions of the registers and bits are given in the subsections that follow.

| Address | Name | | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|---|---|
| 0x0020 | DBGC1 | R | ARM | 0 | XGSBPE | BDM | DBGBRK | | COMRV | |
| | | W | | TRIG | | | | | | |
| 0x0021 | DBGSR | R | TBF | EXTF | 0 | 0 | 0 | SSF2 | SSF1 | SSF0 |
| | | W | | | | | | | | |
| 0x0022 | DBGTCR | R | TSOURCE | | TRANGE | | TRCMOD | | TALIGN | |
| | | W | | | | | | | | |
| 0x0023 | DBGC2 | R | 0 | 0 | 0 | 0 | CDCM | | ABCM | |
| | | W | | | | | | | | |
| 0x0024 | DBGTBH | R | Bit 15 | Bit 14 | Bit 13 | Bit 12 | Bit 11 | Bit 10 | Bit 9 | Bit 8 |
| | | W | | | | | | | | |
| 0x0025 | DBGTBL | R | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
| | | W | | | | | | | | |
| 0x0026 | DBGCNT | R | 0 | CNT | | | | | | |
| | | W | | | | | | | | |
| 0x0027 | DBGSCRX | R | 0 | 0 | 0 | 0 | SC3 | SC2 | SC1 | SC0 |
| | | W | | | | | | | | |
| 0x0027 | DBGMFR | R | 0 | 0 | 0 | 0 | MC3 | MC2 | MC1 | MC0 |
| | | W | | | | | | | | |
| 0x0028[1] | DBGXCTL (COMPA/C) | R | 0 | NDB | TAG | BRK | RW | RWE | SRC | COMPE |
| | | W | | | | | | | | |
| 0x0028[2] | DBGXCTL (COMPB/D) | R | SZE | SZ | TAG | BRK | RW | RWE | SRC | COMPE |
| | | W | | | | | | | | |
| 0x0029 | DBGXAH | R | 0 | Bit 22 | 21 | 20 | 19 | 18 | 17 | Bit 16 |
| | | W | | | | | | | | |
| 0x002A | DBGXAM | R | Bit 15 | 14 | 13 | 12 | 11 | 10 | 9 | Bit 8 |
| | | W | | | | | | | | |
| 0x002B | DBGXAL | R | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
| | | W | | | | | | | | |
| 0x002C | DBGXDH | R | Bit 15 | 14 | 13 | 12 | 11 | 10 | 9 | Bit 8 |
| | | W | | | | | | | | |
| 0x002D | DBGXDL | R | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
| | | W | | | | | | | | |
| 0x002E | DBGXDHM | R | Bit 15 | 14 | 13 | 12 | 11 | 10 | 9 | Bit 8 |
| | | W | | | | | | | | |
| 0x002F | DBGXDLM | R | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
| | | W | | | | | | | | |

[1] This represents the contents if the Comparator A or C control register is blended into this address.

[2] This represents the contents if the Comparator B or D control register is blended into this address

**Figure 8-2. Quick Reference to S12XDBG Registers**

## 8.3.2 Register Descriptions

This section consists of the S12XDBG control and trace buffer register descriptions in address order. Each comparator has a bank of registers that are visible through an 8-byte window between 0x0028 and 0x002F in the S12XDBG module register address map. When ARM is set in DBGC1, the only bits in the S12XDBG module registers that can be written are ARM, TRIG, and COMRV[1:0].

### 8.3.2.1 Debug Control Register 1 (DBGC1)

Address: 0x0020

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | ARM | 0 | XGSBPE | BDM | DBGBRK | | COMRV | |
| W | | TRIG | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 8-3. Debug Control Register (DBGC1)**

Read: Anytime

Write: Bits 7, 1, 0 anytime
Bit 6 can be written anytime but always reads back as 0.
Bits 5:2 anytime S12XDBG is not armed.

**NOTE**

If a write access to DBGC1 with the ARM bit position set occurs simultaneously to a hardware disarm from an internal trigger event, then the ARM bit is cleared due to the hardware disarm.

**NOTE**

When disarming the S12XDBG by clearing ARM with software, the contents of bits[5:2] are not affected by the write, since up until the write operation, ARM = 1 preventing these bits from being written. These bits must be cleared using a second write if required.

**Table 8-5. DBGC1 Field Descriptions**

| Field | Description |
|---|---|
| 7<br>ARM | **Arm Bit** — The ARM bit controls whether the S12XDBG module is armed. This bit can be set and cleared by user software and is automatically cleared on completion of a tracing session, or if a breakpoint is generated with tracing not enabled. On setting this bit the state sequencer enters State1.<br>0 Debugger disarmed<br>1 Debugger armed |
| 6<br>TRIG | **Immediate Trigger Request Bit** — This bit when written to 1 requests an immediate trigger independent of comparator or external tag signal status. When tracing is complete a forced breakpoint may be generated depending upon DBGBRK and BDM bit settings. This bit always reads back a 0. Writing a 0 to this bit has no effect. If TSOURCE are clear no tracing is carried out. If tracing has already commenced using BEGIN- or MID trigger alignment, it continues until the end of the tracing session as defined by the TALIGN bit settings, thus TRIG has no affect. In secure mode tracing is disabled and writing to this bit has no effect.<br>0 Do not trigger until the state sequencer enters the Final State.<br>1 Trigger immediately . |

**Table 8-5. DBGC1 Field Descriptions (continued)**

| Field | Description |
|---|---|
| 5<br>XGSBPE | **XGATE S/W Breakpoint Enable** — The XGSBPE bit controls whether an XGATE S/W breakpoint request is passed to the CPU12X. The XGATE S/W breakpoint request is handled by the S12XDBG module, which can request an CPU12X breakpoint depending on the state of this bit.<br>0 XGATE S/W breakpoint request is disabled<br>1 XGATE S/W breakpoint request is enabled |
| 4<br>BDM | **Background Debug Mode Enable** — This bit determines if an S12X breakpoint causes the system to enter Background Debug Mode (BDM) or initiate a Software Interrupt (SWI). If this bit is set but the BDM is not enabled by the ENBDM bit in the BDM module, then breakpoints default to SWI.<br>0 Breakpoint to Software Interrupt if BDM inactive. Otherwise no breakpoint.<br>1 Breakpoint to BDM, if BDM enabled. Otherwise breakpoint to SWI |
| 3–2<br>DBGBRK | **S12XDBG Breakpoint Enable Bits** — The DBGBRK bits control whether the debugger will request a breakpoint to either CPU12X or XGATE or both upon reaching the state sequencer Final State. If tracing is enabled, the breakpoint is generated on completion of the tracing session. If tracing is not enabled, the breakpoint is generated immediately. Please refer to Section 8.4.7 for further details. XGATE software breakpoints are independent of the DBGBRK bits. XGATE software breakpoints force a breakpoint to the CPU12X independent of the DBGBRK bit field configuration. See Table 8-6. |
| 1–0<br>COMRV | **Comparator Register Visibility Bits** — These bits determine which bank of comparator register is visible in the 8-byte window of the S12XDBG module address map, located between 0x0028 to 0x002F. Furthermore these bits determine which register is visible at the address 0x0027. See Table 8-7. |

**Table 8-6. DBGBRK Encoding**

| DBGBRK | Resource Halted by Breakpoint |
|---|---|
| 00 | No breakpoint generated |
| 01 | XGATE breakpoint generated |
| 10 | CPU12X breakpoint generated |
| 11 | Breakpoints generated for CPU12X and XGATE |

**Table 8-7. COMRV Encoding**

| COMRV | Visible Comparator | Visible Register at 0x0027 |
|---|---|---|
| 00 | Comparator A | DBGSCR1 |
| 01 | Comparator B | DBGSCR2 |
| 10 | Comparator C | DBGSCR3 |
| 11 | Comparator D | DBGMFR |

### 8.3.2.2 Debug Status Register (DBGSR)

Address: 0x0021

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | TBF | EXTF | 0 | 0 | 0 | SSF2 | SSF1 | SSF0 |
| W | | | | | | | | |
| Reset | — | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| POR | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented or Reserved

**Figure 8-4. Debug Status Register (DBGSR)**

Read: Anytime

Write: Never

**Table 8-8. DBGSR Field Descriptions**

| Field | Description |
|---|---|
| 7<br>TBF | **Trace Buffer Full** — The TBF bit indicates that the trace buffer has stored 64 or more lines of data since it was last armed. If this bit is set, then all 64 lines will be valid data, regardless of the value of DBGCNT bits CNT[6:0]. The TBF bit is cleared when ARM in DBGC1 is written to a one. The TBF is cleared by the power on reset initialization. Other system generated resets have no affect on this bit. |
| 6<br>EXTF | **External Tag Hit Flag** — The EXTF bit indicates if a tag hit condition from an external $\overline{\text{TAGHI}}$/$\overline{\text{TAGLO}}$ tag was met since arming. This bit is cleared when ARM in DBGC1 is written to a one.<br>0 External tag hit has not occurred<br>1 External tag hit has occurred |
| 2–0<br>SSF[2:0] | **State Sequencer Flag Bits** — The SSF bits indicate in which state the State Sequencer is currently in. During a debug session on each transition to a new state these bits are updated. If the debug session is ended by software clearing the ARM bit, then these bits retain their value to reflect the last state of the state sequencer before disarming. If a debug session is ended by an internal trigger, then the state sequencer returns to state0 and these bits are cleared to indicate that state0 was entered during the session. On arming the module the state sequencer enters state1 and these bits are forced to SSF[2:0] = 001. See Table 8-9**.** |

**Table 8-9. SSF[2:0] — State Sequence Flag Bit Encoding**

| SSF[2:0] | Current State |
|---|---|
| 000 | State0 (disarmed) |
| 001 | State1 |
| 010 | State2 |
| 011 | State3 |
| 100 | Final State |
| 101,110,111 | Reserved |

### 8.3.2.3 Debug Trace Control Register (DBGTCR)

Address: 0x0022

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | TSOURCE | | TRANGE | | TRCMOD | | TALIGN | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 8-5. Debug Trace Control Register (DBGTCR)**

Read: Anytime

Write: Bits 7:6 only when S12XDBG is neither secure nor armed.
Bits 5:0 anytime the module is disarmed.

**Table 8-10. DBGTCR Field Descriptions**

| Field | Description |
|---|---|
| 7–6<br>TSOURCE | **Trace Source Control Bits** — The TSOURCE bits select the data source for the tracing session. If the MCU system is secured, these bits cannot be set and tracing is inhibited. See Table 8-11. |
| 5–4<br>TRANGE | **Trace Range Bits** — The TRANGE bits allow filtering of trace information from a selected address range when tracing from the CPU12X in Detail Mode. The XGATE tracing range cannot be narrowed using these bits. To use a comparator for range filtering, the corresponding COMPE and SRC bits must remain cleared. If the COMPE bit is not clear then the comparator will also be used to generate state sequence triggers. If the corresponding SRC bit is set the comparator is mapped to the XGATE buses, the TRANGE bits have no effect on the valid address range, memory accesses within the whole memory map are traced. See Table 8-12. |
| 3–2<br>TRCMOD | **Trace Mode Bits** — See Section 8.4.5.2 for detailed Trace Mode descriptions. In Normal Mode, change of flow information is stored. In Loop1 Mode, change of flow information is stored but redundant entries into trace memory are inhibited. In Detail Mode, address and data for all memory and register accesses is stored. See Table 8-13. |
| 1–0<br>TALIGN | **Trigger Align Bits** — These bits control whether the trigger is aligned to the beginning, end or the middle of a tracing session. See Table 8-14. |

**Table 8-11. TSOURCE — Trace Source Bit Encoding**

| TSOURCE | Tracing Source |
|---|---|
| 00 | No tracing requested |
| 01 | CPU12X |
| 10[(1)] | XGATE |
| 11[1,(2)] | Both CPU12X and XGATE |

1. No range limitations are allowed. Thus tracing operates as if TRANGE = 00.
2. No Detail Mode tracing supported. If TRCMOD = 10, no information is stored.

**Table 8-12. TRANGE Trace Range Encoding**

| TRANGE | Tracing Range |
|---|---|
| 00 | Trace from all addresses (No filter) |
| 01 | Trace only in address range from $00000 to Comparator D |
| 10 | Trace only in address range from Comparator C to $7FFFFF |
| 11 | Trace only in range from Comparator C to Comparator D |

**Table 8-13. TRCMOD Trace Mode Bit Encoding**

| TRCMOD | Description |
|---|---|
| 00 | Normal |
| 01 | Loop1 |
| 10 | Detail |
| 11 | Pure PC |

**Table 8-14. TALIGN Trace Alignment Encoding**

| TALIGN | Description |
|---|---|
| 00 | Trigger at end of stored data |
| 01 | Trigger before storing data |
| 10 | Trace buffer entries before and after trigger |
| 11 | Reserved |

### 8.3.2.4 Debug Control Register2 (DBGC2)

Address: 0x0023

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | 0 | 0 | CDCM | | ABCM | |
| W | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented or Reserved

**Figure 8-6. Debug Control Register2 (DBGC2)**

Read: Anytime

Write: Anytime the module is disarmed.

This register configures the comparators for range matching.

**Table 8-15. DBGC2 Field Descriptions**

| Field | Description |
|---|---|
| 3–2<br>CDCM[1:0] | **C and D Comparator Match Control** — These bits determine the C and D comparator match mapping as described in Table 8-16. |
| 1–0<br>ABCM[1:0] | **A and B Comparator Match Control** — These bits determine the A and B comparator match mapping as described in Table 8-17. |

**Table 8-16. CDCM Encoding**

| CDCM | Description |
|---|---|
| 00 | Match2 mapped to comparator C match..... Match3 mapped to comparator D match. |
| 01 | Match2 mapped to comparator C/D inside range..... Match3 disabled. |
| 10 | Match2 mapped to comparator C/D outside range..... Match3 disabled. |
| 11 | Reserved(1) |

1. Currently defaults to Match2 mapped to comparator C : Match3 mapped to comparator D

**Table 8-17. ABCM Encoding**

| ABCM | Description |
|---|---|
| 00 | Match0 mapped to comparator A match..... Match1 mapped to comparator B match. |
| 01 | Match 0 mapped to comparator A/B inside range..... Match1 disabled. |
| 10 | Match 0 mapped to comparator A/B outside range..... Match1 disabled. |
| 11 | Reserved(1) |

1. Currently defaults to Match0 mapped to comparator A : Match1 mapped to comparator B

### 8.3.2.5 Debug Trace Buffer Register (DBGTBH:DBGTBL)

Address: 0x0024, 0x0025

| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | Bit 15 | Bit 14 | Bit 13 | Bit 12 | Bit 11 | Bit 10 | Bit 9 | Bit 8 | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
| POR | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X |
| Other Resets | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |

**Figure 8-7. Debug Trace Buffer Register (DBGTB)**

Read: Only when unlocked AND not secured AND not armed AND with a TSOURCE bit set.

Write: Aligned word writes when disarmed unlock the trace buffer for reading but do not affect trace buffer contents.

**Table 8-18. DBGTB Field Descriptions**

| Field | Description |
|---|---|
| 15–0<br>Bit[15:0] | **Trace Buffer Data Bits** — The Trace Buffer Register is a window through which the 64-bit wide data lines of the Trace Buffer may be read 16 bits at a time. Each valid read of DBGTB increments an internal trace buffer pointer which points to the next address to be read. When the ARM bit is written to 1 the trace buffer is locked to prevent reading. The trace buffer can only be unlocked for reading by writing to DBGTB with an aligned word write when the module is disarmed. The DBGTB register can be read only as an aligned word, any byte reads or misaligned access of these registers will return 0 and will not cause the trace buffer pointer to increment to the next trace buffer address. The same is true for word reads while the debugger is armed. The POR state is undefined Other resets do not affect the trace buffer contents. . |

### 8.3.2.6 Debug Count Register (DBGCNT)

Address: 0x0026

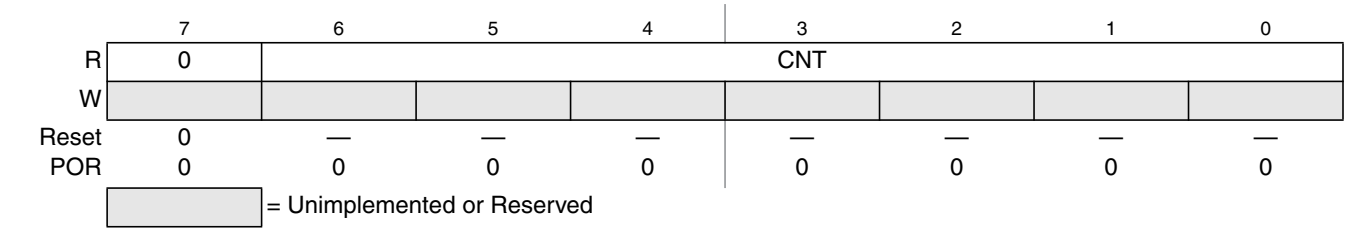



**Figure 8-8. Debug Count Register (DBGCNT)**

Read: Anytime

Write: Never

**Table 8-19. DBGCNT Field Descriptions**

| Field | Description |
|---|---|
| 6–0<br>CNT[6:0] | **Count Value** — The CNT bits [6:0] indicate the number of valid data 64-bit data lines stored in the Trace Buffer. Table 8-20 shows the correlation between the CNT bits and the number of valid data lines in the Trace Buffer. When the CNT rolls over to zero, the TBF bit in DBGSR is set and incrementing of CNT will continue in end-trigger or mid-trigger mode. The DBGCNT register is cleared when ARM in DBGC1 is written to a one. The DBGCNT register is cleared by power-on-reset initialization but is not cleared by other system resets. Thus should a reset occur during a debug session, the DBGCNT register still indicates after the reset, the number of valid trace buffer entries stored before the reset occurred. The DBGCNT register is not decremented when reading from the trace buffer. |

**Table 8-20. CNT Decoding Table**

| TBF (DBGSR) | CNT[6:0] | Description |
|---|---|---|
| 0 | 0000000 | No data valid |
| 0 | 0000001 | 32 bits of one line valid(1) |
| 0 | 0000010<br>0000100<br>0000110<br>..<br>1111100 | 1 line valid<br>2 lines valid<br>3 lines valid<br>..<br>62 lines valid |
| 0 | 1111110 | 63 lines valid |
| 1 | 0000000 | 64 lines valid; if using Begin trigger alignment,<br>ARM bit will be cleared and the tracing session ends. |
| 1 | 0000010<br>..<br>..<br>1111110 | 64 lines valid,<br>oldest data has been overwritten by most recent data |

1. This applies to Normal/Loop1/PurePC Modes when tracing from either CPU12X or XGATE only.

### 8.3.2.7 Debug State Control Registers

There is a dedicated control register for each of the state sequencer states 1 to 3 that determines if transitions from that state are allowed, depending upon comparator matches or tag hits, and defines the next state for the state sequencer following a match. The three debug state control registers are located at the same address in the register address map (0x0027). Each register can be accessed using the COMRV bits in DBGC1 to blend in the required register. The COMRV = 11 value blends in the match flag register (DBGMFR).

**Table 8-21. State Control Register Access Encoding**

| COMRV | Visible State Control Register |
|---|---|
| 00 | DBGSCR1 |
| 01 | DBGSCR2 |
| 10 | DBGSCR3 |
| 11 | DBGMFR |

#### 8.3.2.7.1 Debug State Control Register 1 (DBGSCR1)

Address: 0x0027

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | 0 | 0 | SC3 | SC2 | SC1 | SC0 |
| W | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented or Reserved

**Figure 8-9. Debug State Control Register 1 (DBGSCR1)**

Read: If COMRV[1:0] = 00

Write: If COMRV[1:0] = 00 and S12XDBG is not armed.

This register is visible at 0x0027 only with COMRV[1:0] = 00. The state control register 1 selects the targeted next state whilst in State1. The matches refer to the match channels of the comparator match control logic as depicted in Figure 8-1 and described in Section 8.3.2.8.1. Comparators must be enabled by setting the comparator enable bit in the associated DBGXCTL control register.

**Table 8-22. DBGSCR1 Field Descriptions**

| Field | Description |
|---|---|
| 3–0<br>SC[3:0] | These bits select the targeted next state whilst in State1, based upon the match event. |

**Table 8-23. State1 Sequencer Next State Selection**

| SC[3:0] | Description |
|---|---|
| 0000 | Any match triggers to state2 |
| 0001 | Any match triggers to state3 |
| 0010 | Any match triggers to Final State |

**Table 8-23. State1 Sequencer Next State Selection (continued)**

| SC[3:0] | Description |
|---|---|
| 0011 | Match2 triggers to State2..... Other matches have no effect |
| 0100 | Match2 triggers to State3..... Other matches have no effect |
| 0101 | Match2 triggers to Final State..... Other matches have no effect |
| 0110 | Match0 triggers to State2..... Match1 triggers to State3..... Other matches have no effect |
| 0111 | Match1 triggers to State3..... Match0 triggers Final State..... Other matches have no effect |
| 1000 | Match0 triggers to State2..... Match2 triggers to State3..... Other matches have no effect |
| 1001 | Match2 triggers to State3..... Match0 triggers Final State..... Other matches have no effect |
| 1010 | Match1 triggers to State2..... Match3 triggers to State3..... Other matches have no effect |
| 1011 | Match3 triggers to State3..... Match1 triggers to Final State..... Other matches have no effect |
| 1100 | Match3 has no effect..... All other matches (M0,M1,M2) trigger to State2 |
| 1101 | Reserved. (No match triggers state sequencer transition) |
| 1110 | Reserved. (No match triggers state sequencer transition) |
| 1111 | Reserved. (No match triggers state sequencer transition) |

The trigger priorities described in Table 8-42 dictate that in the case of simultaneous matches, the match on the lower channel number (0,1,2,3) has priority. The SC[3:0] encoding ensures that a match leading to final state has priority over all other matches.

### 8.3.2.7.2 Debug State Control Register 2 (DBGSCR2)

Address: 0x0027

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | 0 | 0 | SC3 | SC2 | SC1 | SC0 |
| W | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented or Reserved

**Figure 8-10. Debug State Control Register 2 (DBGSCR2)**

Read: If COMRV[1:0] = 01

Write: If COMRV[1:0] = 01 and S12XDBG is not armed.

This register is visible at 0x0027 only with COMRV[1:0] = 01. The state control register 2 selects the targeted next state whilst in State2. The matches refer to the match channels of the comparator match control logic as depicted in Figure 8-1 and described in Section 8.3.2.8.1. Comparators must be enabled by setting the comparator enable bit in the associated DBGXCTL control register.

**Table 8-24. DBGSCR2 Field Descriptions**

| Field | Description |
|---|---|
| 3–0<br>SC[3:0] | These bits select the targeted next state whilst in State2, based upon the match event. |

**Table 8-25. State2 —Sequencer Next State Selection**

| SC[3:0] | Description |
|---|---|
| 0000 | Any match triggers to state1 |

**Table 8-25. State2 —Sequencer Next State Selection (continued)**

| SC[3:0] | Description |
|---|---|
| 0001 | Any match triggers to state3 |
| 0010 | Any match triggers to Final State |
| 0011 | Match3 triggers to State1..... Other matches have no effect |
| 0100 | Match3 triggers to State3..... Other matches have no effect |
| 0101 | Match3 triggers to Final State..... Other matches have no effect |
| 0110 | Match0 triggers to State1..... Match1 triggers to State3..... Other matches have no effect |
| 0111 | Match1 triggers to State3..... Match0 triggers Final State..... Other matches have no effect |
| 1000 | Match0 triggers to State1..... Match2 triggers to State3..... Other matches have no effect |
| 1001 | Match2 triggers to State3..... Match0 triggers Final State..... Other matches have no effect |
| 1010 | Match1 triggers to State1..... Match3 triggers to State3..... Other matches have no effect |
| 1011 | Match3 triggers to State3..... Match1 triggers Final State..... Other matches have no effect |
| 1100 | Match2 triggers to State1..... Match3 trigger to Final State |
| 1101 | Match2 has no affect, all other matches (M0,M1,M3) trigger to Final State |
| 1110 | Reserved. (No match triggers state sequencer transition) |
| 1111 | Reserved. (No match triggers state sequencer transition) |

The trigger priorities described in Table 8-42 dictate that in the case of simultaneous matches, the match on the lower channel number (0,1,2,3) has priority. The SC[3:0] encoding ensures that a match leading to final state has priority over all other matches.

### 8.3.2.7.3 Debug State Control Register 3 (DBGSCR3)

Address: 0x0027

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | 0 | 0 | SC3 | SC2 | SC1 | SC0 |
| W | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented or Reserved

**Figure 8-11. Debug State Control Register 3 (DBGSCR3)**

Read: If COMRV[1:0] = 10

Write: If COMRV[1:0] = 10 and S12XDBG is not armed.

This register is visible at 0x0027 only with COMRV[1:0] = 10. The state control register three selects the targeted next state whilst in State3. The matches refer to the match channels of the comparator match control logic as depicted in Figure 8-1 and described in Section 8.3.2.8.1. Comparators must be enabled by setting the comparator enable bit in the associated DBGXCTL control register.

**Table 8-26. DBGSCR3 Field Descriptions**

| Field | Description |
|---|---|
| 3–0<br>SC[3:0] | These bits select the targeted next state whilst in State3, based upon the match event. |

**Table 8-27. State3 — Sequencer Next State Selection**

| SC[3:0] | Description |
|---|---|
| 0000 | Any match triggers to state1 |
| 0001 | Any match triggers to state2 |
| 0010 | Any match triggers to Final State |
| 0011 | Match0 triggers to State1..... Other matches have no effect |
| 0100 | Match0 triggers to State2..... Other matches have no effect |
| 0101 | Match0 triggers to Final State.....Match1 triggers to State1...Other matches have no effect |
| 0110 | Match1 triggers to State1..... Other matches have no effect |
| 0111 | Match1 triggers to State2..... Other matches have no effect |
| 1000 | Match1 triggers to Final State..... Other matches have no effect |
| 1001 | Match2 triggers to State2..... Match0 triggers to Final State..... Other matches have no effect |
| 1010 | Match1 triggers to State1..... Match3 triggers to State2..... Other matches have no effect |
| 1011 | Match3 triggers to State2..... Match1 triggers to Final State..... Other matches have no effect |
| 1100 | Match2 triggers to Final State..... Other matches have no effect |
| 1101 | Match3 triggers to Final State..... Other matches have no effect |
| 1110 | Reserved. (No match triggers state sequencer transition) |
| 1111 | Reserved. (No match triggers state sequencer transition) |

The trigger priorities described in Table 8-42 dictate that in the case of simultaneous matches, the match on the lower channel number (0,1,2,3) has priority. The SC[3:0] encoding ensures that a match leading to final state has priority over all other matches.

### 8.3.2.7.4 Debug Match Flag Register (DBGMFR)

Address: 0x0027

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | 0 | 0 | MC3 | MC2 | MC1 | MC0 |
| W | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented or Reserved

**Figure 8-12. Debug Match Flag Register (DBGMFR)**

Read: If COMRV[1:0] = 11

Write: Never

DBGMFR is visible at 0x0027 only with COMRV[1:0] = 11. It features four flag bits each mapped directly to a channel. Should a match occur on the channel during the debug session, then the corresponding flag is set and remains set until the next time the module is armed by writing to the ARM bit. Thus the contents are retained after a debug session for evaluation purposes. These flags cannot be cleared by software, they are cleared only when arming the module. A set flag does not inhibit the setting of other flags. Once a flag is set, further triggers on the same channel have no affect.

### 8.3.2.8 Comparator Register Descriptions

Each comparator has a bank of registers that are visible through an 8-byte window in the S12XDBG module register address map. Comparators A and C consist of 8 register bytes (3 address bus compare registers, two data bus compare registers, two data bus mask registers and a control register).

Comparators B and D consist of four register bytes (three address bus compare registers and a control register).

Each set of comparator registers is accessible in the same 8-byte window of the register address map and can be accessed using the COMRV bits in the DBGC1 register. If the Comparators B or D are accessed through the 8-byte window, then only the address and control bytes are visible, the 4 bytes associated with data bus and data bus masking read as zero and cannot be written. Furthermore the control registers for comparators B and D differ from those of comparators A and C.

**Table 8-28. Comparator Register Layout**

| | | | |
|---|---|---|---|
| 0x0028 | CONTROL | Read/Write | Comparators A,B,C,D |
| 0x0029 | ADDRESS HIGH | Read/Write | Comparators A,B,C,D |
| 0x002A | ADDRESS MEDIUM | Read/Write | Comparators A,B,C,D |
| 0x002B | ADDRESS LOW | Read/Write | Comparators A,B,C,D |
| 0x002C | DATA HIGH COMPARATOR | Read/Write | Comparator A and C only |
| 0x002D | DATA LOW COMPARATOR | Read/Write | Comparator A and C only |
| 0x002E | DATA HIGH MASK | Read/Write | Comparator A and C only |
| 0x002F | DATA LOW MASK | Read/Write | Comparator A and C only |

#### 8.3.2.8.1 Debug Comparator Control Register (DBGXCTL)

The contents of this register bits 7 and 6 differ depending upon which comparator registers are visible in the 8-byte window of the DBG module register address map.

Address: 0x0028

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | 0 | NDB | TAG | BRK | RW | RWE | SRC | COMPE |
| W | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented or Reserved

**Figure 8-13. Debug Comparator Control Register (Comparators A and C)**

Address: 0x0028

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | SZE | SZ | TAG | BRK | RW | RWE | SRC | COMPE |
| W | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 8-14. Debug Comparator Control Register (Comparators B and D)**

Read: Anytime. See Table 8-29 for visible register encoding.

Write: If DBG not armed. See Table 8-29 for visible register encoding.

The DBGC1_COMRV bits determine which comparator control, address, data and datamask registers are visible in the 8-byte window from 0x0028 to 0x002F as shown in Section Table 8-29.

**Table 8-29. Comparator Address Register Visibility**

| COMRV | Visible Comparator |
|---|---|
| 00 | DBGACTL, DBGAAH ,DBGAAM, DBGAAL, DBGADH, DBGADL, DBGADHM, DBGADLM |
| 01 | DBGBCTL, DBGBAH, DBGBAM, DBGBAL |
| 10 | DBGCCTL, DBGCAH, DBGCAM, DBGCAL, DBGCDH, DBGCDL, DBGCDHM, DBGCDLM |
| 11 | DBGDCTL, DBGDAH, DBGDAM, DBGDAL |

**Table 8-30. DBGXCTL Field Descriptions**

| Field | Description |
|---|---|
| 7<br>SZE<br>(Comparators B and D) | **Size Comparator Enable Bit** — The SZE bit controls whether access size comparison is enabled for the associated comparator. This bit is ignored if the TAG bit in the same register is set.<br>0 Word/Byte access size is not used in comparison<br>1 Word/Byte access size is used in comparison |
| 6<br>NDB<br>(Comparators A and C | **Not Data Bus** — The NDB bit controls whether the match occurs when the data bus matches the comparator register value or when the data bus differs from the register value. Furthermore data bus bits can be individually masked using the comparator data mask registers. This bit is only available for comparators A and C. This bit is ignored if the TAG bit in the same register is set. This bit position has an SZ functionality for comparators B and D.<br>0 Match on data bus equivalence to comparator register contents<br>1 Match on data bus difference to comparator register contents |
| 6<br>SZ<br>(Comparators B and D) | **Size Comparator Value Bit** — The SZ bit selects either word or byte access size in comparison for the associated comparator. This bit is ignored if the SZE bit is cleared or if the TAG bit in the same register is set. This bit position has NDB functionality for comparators A and C<br>0 Word access size will be compared<br>1 Byte access size will be compared |
| 5<br>TAG | **Tag Select** — This bit controls whether the comparator match will cause a trigger or tag the opcode at the matched address. Tagged opcodes trigger only if they reach the execution stage of the instruction queue.<br>0 Trigger immediately on match<br>1 On match, tag the opcode. If the opcode is about to be executed a trigger is generated |
| 4<br>BRK | **Break** — This bit controls whether a channel match terminates a debug session immediately, independent of state sequencer state. To generate an immediate breakpoint the module breakpoints must be enabled using DBGBRK.<br>0 The debug session termination is dependent upon the state sequencer and trigger conditions.<br>1 A match on this channel terminates the debug session immediately; breakpoints if active are generated, tracing, if active, is terminated and the module disarmed. |
| 3<br>RW | **Read/Write Comparator Value Bit** — The RW bit controls whether read or write is used in compare for the associated comparator . The RW bit is not used if RWE = 0.<br>0 Write cycle will be matched<br>1 Read cycle will be matched |
| 2<br>RWE | **Read/Write Enable Bit** — The RWE bit controls whether read or write comparison is enabled for the associated comparator. This bit is not used for tagged operations.<br>0 Read/Write is not used in comparison<br>1 Read/Write is used in comparison |

**Table 8-30. DBGXCTL Field Descriptions (continued)**

| Field | Description |
|---|---|
| 1<br>SRC | Determines mapping of comparator to CPU12X or XGATE<br>0 The comparator is mapped to CPU12X buses<br>1 The comparator is mapped to XGATE address and data buses |
| 0<br>COMPE | Determines if comparator is enabled<br>0 The comparator is not enabled<br>1 The comparator is enabled for state sequence triggers or tag generation |

Table 8-31 shows the effect for RWE and RW on the comparison conditions. These bits are not useful for tagged operations since the trigger occurs based on the tagged opcode reaching the execution stage of the instruction queue. Thus these bits are ignored if tagged triggering is selected.

**Table 8-31. Read or Write Comparison Logic Table**

| RWE Bit | RW Bit | RW Signal | Comment |
|---|---|---|---|
| 0 | x | 0 | RW not used in comparison |
| 0 | x | 1 | RW not used in comparison |
| 1 | 0 | 0 | Write |
| 1 | 0 | 1 | No match |
| 1 | 1 | 0 | No match |
| 1 | 1 | 1 | Read |

### 8.3.2.8.2 Debug Comparator Address High Register (DBGXAH)

Address: 0x0029

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | 0 | Bit 22 | Bit 21 | Bit 20 | Bit 19 | Bit 18 | Bit 17 | Bit 16 |
| W | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented or Reserved

**Figure 8-15. Debug Comparator Address High Register (DBGXAH)**

Read: Anytime. See Table 8-29 for visible register encoding.

Write: If DBG not armed. See Table 8-29 for visible register encoding.

**Table 8-32. DBGXAH Field Descriptions**

| Field | Description |
|---|---|
| 6–0<br>Bit[22:16] | **Comparator Address High Compare Bits** — The Comparator address high compare bits control whether the selected comparator will compare the address bus bits [22:16] to a logic one or logic zero. This register byte is ignored for XGATE compares.<br>0 Compare corresponding address bit to a logic zero<br>1 Compare corresponding address bit to a logic one |

### 8.3.2.8.3 Debug Comparator Address Mid Register (DBGXAM)

Address: 0x002A

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | Bit 15 | Bit 14 | Bit 13 | Bit 12 | Bit 11 | Bit 10 | Bit 9 | Bit 8 |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 8-16. Debug Comparator Address Mid Register (DBGXAM)**

Read: Anytime. See Table 8-29 for visible register encoding.

Write: If DBG not armed. See Table 8-29 for visible register encoding.

**Table 8-33. DBGXAM Field Descriptions**

| Field | Description |
|---|---|
| 7–0<br>Bit[15:8] | **Comparator Address Mid Compare Bits**— The Comparator address mid compare bits control whether the selected comparator will compare the address bus bits [15:8] to a logic one or logic zero.<br>0 Compare corresponding address bit to a logic zero<br>1 Compare corresponding address bit to a logic one |

### 8.3.2.8.4 Debug Comparator Address Low Register (DBGXAL)

Address: 0x002B

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 8-17. Debug Comparator Address Low Register (DBGXAL)**

Read: Anytime. See Table 8-29 for visible register encoding.

Write: If DBG not armed. See Table 8-29 for visible register encoding.

**Table 8-34. DBGXAL Field Descriptions**

| Field | Description |
|---|---|
| 7–0<br>Bits[7:0] | **Comparator Address Low Compare Bits** — The Comparator address low compare bits control whether the selected comparator will compare the address bus bits [7:0] to a logic one or logic zero.<br>0 Compare corresponding address bit to a logic zero<br>1 Compare corresponding address bit to a logic one |

### 8.3.2.8.5 Debug Comparator Data High Register (DBGXDH)

Address: 0x002C

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | Bit 15 | Bit 14 | Bit 13 | Bit 12 | Bit 11 | Bit 10 | Bit 9 | Bit 8 |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 8-18. Debug Comparator Data High Register (DBGXDH)**

Read: Anytime. See Table 8-29 for visible register encoding.

Write: If DBG not armed. See Table 8-29 for visible register encoding.

**Table 8-35. DBGXAH Field Descriptions**

| Field | Description |
|---|---|
| 7–0<br>Bits[15:8] | **Comparator Data High Compare Bits** — The Comparator data high compare bits control whether the selected comparator compares the data bus bits [15:8] to a logic one or logic zero. The comparator data compare bits are only used in comparison if the corresponding data mask bit is logic 1. This register is available only for comparators A and C.<br>0 Compare corresponding data bit to a logic zero<br>1 Compare corresponding data bit to a logic one |

### 8.3.2.8.6 Debug Comparator Data Low Register (DBGXDL)

Address: 0x002D

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 8-19. Debug Comparator Data Low Register (DBGXDL)**

Read: Anytime. See Table 8-29 for visible register encoding.

Write: If DBG not armed. See Table 8-29 for visible register encoding.

**Table 8-36. DBGXDL Field Descriptions**

| Field | Description |
|---|---|
| 7–0<br>Bits[7:0] | **Comparator Data Low Compare Bits** — The Comparator data low compare bits control whether the selected comparator compares the data bus bits [7:0] to a logic one or logic zero. The comparator data compare bits are only used in comparison if the corresponding data mask bit is logic 1. This register is available only for comparators A and C.<br>0 Compare corresponding data bit to a logic zero<br>1 Compare corresponding data bit to a logic one |

### 8.3.2.8.7 Debug Comparator Data High Mask Register (DBGXDHM)

Address: 0x002E

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | Bit 15 | Bit 14 | Bit 13 | Bit 12 | Bit 11 | Bit 10 | Bit 9 | Bit 8 |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 8-20. Debug Comparator Data High Mask Register (DBGXDHM)**

Read: Anytime. See Table 8-29 for visible register encoding.

Write: If DBG not armed. See Table 8-29 for visible register encoding.

**Table 8-37. DBGXDHM Field Descriptions**

| Field | Description |
|---|---|
| 7–0<br>Bits[15:8] | **Comparator Data High Mask Bits** — The Comparator data high mask bits control whether the selected comparator compares the data bus bits [15:8] to the corresponding comparator data compare bits. This register is available only for comparators A and C.<br>0 Do not compare corresponding data bit<br>1 Compare corresponding data bit |

### 8.3.2.8.8 Debug Comparator Data Low Mask Register (DBGXDLM)

Address: 0x002F

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 8-21. Debug Comparator Data Low Mask Register (DBGXDLM)**

Read: Anytime. See Table 8-29 for visible register encoding.

Write: If DBG not armed. See Table 8-29 for visible register encoding.

**Table 8-38. DBGXDLM Field Descriptions**

| Field | Description |
|---|---|
| 7–0<br>Bits[7:0] | **Comparator Data Low Mask Bits** — The Comparator data low mask bits control whether the selected comparator compares the data bus bits [7:0] to the corresponding comparator data compare bits. This register is available only for comparators A and C.<br>0 Do not compare corresponding data bit<br>1 Compare corresponding data bit |

## 8.4 Functional Description

This section provides a complete functional description of the S12XDBG module. If the part is in secure mode, the S12XDBG module can generate breakpoints but tracing is not possible.

### 8.4.1 S12XDBG Operation

Arming the S12XDBG module by setting ARM in DBGC1 allows triggering, and storing of data in the trace buffer and can be used to cause breakpoints to the CPU12X or the XGATE module. The DBG module is made up of four main blocks, the comparators, control logic, the state sequencer, and the trace buffer.

The comparators monitor the bus activity of the CPU12X and XGATE. Comparators can be configured to monitor address and databus. Comparators can also be configured to mask out individual data bus bits during a compare and to use R/W and word/byte access qualification in the comparison. When a match with a comparator register value occurs the associated control logic can trigger the state sequencer to another state (see Figure 8-22). Either forced or tagged triggers are possible. Using a forced trigger, the trigger is generated immediately on a comparator match. Using a tagged trigger, at a comparator match, the instruction opcode is tagged and only if the instruction reaches the execution stage of the instruction queue is a trigger generated. In the case of a transition to Final State, bus tracing is triggered and/or a breakpoint can be generated. Tracing of both CPU12X and/or XGATE bus activity is possible.

Independent of the state sequencer, a breakpoint can be triggered by the external $\overline{\text{TAGHI}}$ / $\overline{\text{TAGLO}}$ signals or by an XGATE S/W breakpoint request or by writing to the TRIG bit in the DBGC1 control register.

The trace buffer is visible through a 2-byte window in the register address map and can be read out using standard 16-bit word reads.

### 8.4.2 Comparator Modes

The S12XDBG contains four comparators, A, B, C, and D. Each comparator can be configured to monitor CPU12X or XGATE buses. Each comparator compares the selected address bus with the address stored in DBGXAH, DBGXAM, and DBGXAL. Furthermore, comparators A and C also compare the data buses to the data stored in DBGXDH, DBGXDL and allow masking of individual data bus bits.

S12X comparator matches are disabled in BDM and during BDM accesses.

The comparator match control logic configures comparators to monitor the buses for an exact address or an address range. The comparator configuration is controlled by the control register contents and the range control by the DBGC2 contents.

On a match a trigger can initiate a transition to another state sequencer state (see Section 8.4.3”). The comparator control register also allows the type of access to be included in the comparison through the use of the RWE, RW, SZE, and SZ bits. The RWE bit controls whether read or write comparison is enabled for the associated comparator and the RW bit selects either a read or write access for a valid match. Similarly the SZE and SZ bits allows the size of access (word or byte) to be considered in the compare. Only comparators B and D feature SZE and SZ.

The TAG bit in each comparator control register is used to determine the triggering condition. By setting TAG, the comparator will qualify a match with the output of opcode tracking logic and a trigger occurs before the tagged instruction executes (tagged-type trigger). Whilst tagging, the RW, RWE, SZE, and SZ bits are ignored and the comparator register must be loaded with the exact opcode address.

If the TAG bit is clear (forced type trigger) a comparator match is generated when the selected address appears on the system address bus. If the selected address is an opcode address, the match is generated

when the opcode is fetched from the memory. This precedes the instruction execution by an indefinite number of cycles due to instruction pipe lining. For a comparator match of an opcode at an odd address when TAG = 0, the corresponding even address must be contained in the comparator register. Thus for an opcode at odd address (n), the comparator register must contain address (n–1).

Once a successful comparator match has occurred, the condition that caused the original match is not verified again on subsequent matches. Thus if a particular data value is verified at a given address, this address may not still contain that data value when a subsequent match occurs.

Comparators C and D can also be used to select an address range to trace from. This is determined by the TRANGE bits in the DBGTCR register. The TRANGE encoding is shown in Table 8-12. If the TRANGE bits select a range definition using comparator D, then comparator D is configured for trace range definition and cannot be used for address bus comparisons. Similarly if the TRANGE bits select a range definition using comparator C, then comparator C is configured for trace range definition and cannot be used for address bus comparisons.

Match[0, 1, 2, 3] map directly to Comparators[A, B, C, D] respectively, except in range modes (see Section 8.3.2.4). Comparator priority rules are described in the trigger priority section (Section 8.4.3.6).

### 8.4.2.1 Exact Address Comparator Match (Comparators A and C)

With range comparisons disabled, the match condition is an exact equivalence of address/data bus with the value stored in the comparator address/data registers. Further qualification of the type of access (R/W, word/byte) is possible.

Comparators A and C do not feature SZE or SZ control bits, thus the access size is not compared. Table 8-40 lists access considerations without data bus compare. Table 8-39 lists access considerations with data bus comparison. To compare byte accesses DBGxDH must be loaded with the data byte, the low byte must be masked out using the DBGxDLM mask register. On word accesses the data byte of the lower address is mapped to DBGxDH.

**Table 8-39. Comparator A and C Data Bus Considerations**

| Access | Address | DBGxDH | DBGxDL | DBGxDHM | DBGxDLM | Example Valid Match | |
|---|---|---|---|---|---|---|---|
| Word | ADDR[n] | Data[n] | Data[n+1] | $FF | $FF | MOVW #$WORD ADDR[n] | config1 |
| Byte | ADDR[n] | Data[n] | x | $FF | $00 | MOVB #$BYTE ADDR[n] | config2 |
| Word | ADDR[n] | Data[n] | x | $FF | $00 | MOVW #$WORD ADDR[n] | config2 |
| Word | ADDR[n] | x | Data[n+1] | $00 | $FF | MOVW #$WORD ADDR[n] | config3 |

Code may contain various access forms of the same address, i.e. a word access of ADDR[n] or byte access of ADDR[n+1] both access n+1. At a word access of ADDR[n], address ADDR[n+1] does not appear on the address bus and so cannot cause a comparator match if the comparator contains ADDR[n]. Thus it is not possible to monitor all data accesses of ADDR[n+1] with one comparator.

To detect an access of ADDR[n+1] through a word access of ADDR[n] the comparator can be configured to ADDR[n], DBGxDL is loaded with the data pattern and DBGxDHM is cleared so only the data[n+1] is compared on accesses of ADDR[n].

**NOTE**

Using this configuration, a byte access of ADDR[n] can cause a comparator match if the databus low byte by chance contains the same value as ADDR[n+1] because the databus comparator does not feature access size comparison and uses the mask as a “don’t care” function. Thus masked bits do not prevent a match.

Comparators A and C feature an NDB control bit to determine if a match occurs when the data bus differs to comparator register contents or when the data bus is equivalent to the comparator register contents.

### 8.4.2.2 Exact Address Comparator Match (Comparators B and D)

Comparators B and D feature SZ and SZE control bits. If SZE is clear, then the comparator address match qualification functions the same as for comparators A and C.

If the SZE bit is set the access size (word or byte) is compared with the SZ bit value such that only the specified type of access causes a match. Thus if configured for a byte access of a particular address, a word access covering the same address does not lead to match.

**Table 8-40. Comparator Access Size Considerations**

| Comparator | Address | SZE | SZ8 | Condition For Valid Match |
|---|---|---|---|---|
| Comparators A and C | ADDR[n] | — | — | Word and byte accesses of ADDR[n](1)<br>MOVB #$BYTE ADDR[n]<br>MOVW #$WORD ADDR[n] |
| Comparators B and D | ADDR[n] | 0 | X | Word and byte accesses of ADDR[n]1<br>MOVB #$BYTE ADDR[n]<br>MOVW #$WORD ADDR[n] |
| Comparators B and D | ADDR[n] | 1 | 0 | Word accesses of ADDR[n]1<br>MOVW #$WORD ADDR[n] |
| Comparators B and D | ADDR[n] | 1 | 1 | Byte accesses of ADDR[n]<br>MOVB #$BYTE ADDR[n] |

1. A word access of ADDR[n-1] also accesses ADDR[n] but does not generate a match.
The comparator address register must contain the exact address used in the code.

### 8.4.2.3 Data Bus Comparison NDB Dependency

Comparators A and C each feature an NDB control bit, which allows data bus comparators to be configured to either trigger on equivalence or trigger on difference. This allows monitoring of a difference in the contents of an address location from an expected value.

When matching on an equivalence (NDB=0), each individual data bus bit position can be masked out by clearing the corresponding mask bit (DBGxDHM/DBGxDLM), so that it is ignored in the comparison. A match occurs when all data bus bits with corresponding mask bits set are equivalent. If all mask register bits are clear, then a match is based on the address bus only, the data bus is ignored.

When matching on a difference, mask bits can be cleared to ignore bit positions. A match occurs when any data bus bit with corresponding mask bit set is different. Clearing all mask bits, causes all bits to be ignored and prevents a match because no difference can be detected. In this case address bus equivalence does not cause a match.

**Table 8-41. NDB and MASK bit dependency**

| NDB | DBGxDHM[n] / DBGxDLM[n] | Comment |
|---|---|---|
| 0 | 0 | Do not compare data bus bit. |
| 0 | 1 | Compare data bus bit. Match on equivalence. |
| 1 | 0 | Do not compare data bus bit. |
| 1 | 1 | Compare data bus bit. Match on difference. |

### 8.4.2.4 Range Comparisons

When using the AB comparator pair for a range comparison, the data bus can also be used for qualification by using the comparator A data and data mask registers. Furthermore the DBGACTL RW and RWE bits can be used to qualify the range comparison on either a read or a write access. The corresponding DBGBCTL bits are ignored. Similarly when using the CD comparator pair for a range comparison, the data bus can also be used for qualification by using the comparator C data and data mask registers. Furthermore the DBGCCTL RW and RWE bits can be used to qualify the range comparison on either a read or a write access if tagging is not selected. The corresponding DBGDCTL bits are ignored. The SZE and SZ control bits are ignored in range mode. The comparator A and C TAG bits are used to tag range comparisons for the AB and CD ranges respectively. The comparator B and D TAG bits are ignored in range modes. In order for a range comparison using comparators A and B, both COMPEA and COMPEB must be set; to disable range comparisons both must be cleared. Similarly for a range CD comparison, both COMPEC and COMPED must be set. If a range mode is selected SRCA and SRCC select the source (S12X or XGATE), SRCB and SRCD are ignored. The comparator A and C BRK bits are used for the AB and CD ranges respectively, the comparator B and D BRK bits are ignored in range mode. When configured for range comparisons and tagging, the ranges are accurate only to word boundaries.

#### 8.4.2.4.1 Inside Range (CompAC_Addr $\leq$ address $\leq$ CompBD_Addr)

In the Inside Range comparator mode, either comparator pair A and B or comparator pair C and D can be configured for range comparisons by the control register (DBGC2). The match condition requires that a valid match for both comparators happens on the same bus cycle. A match condition on only one comparator is not valid. An aligned word access which straddles the range boundary will cause a trigger only if the aligned address is inside the range.

#### 8.4.2.4.2 Outside Range (address < CompAC_Addr or address > CompBD_Addr)

In the Outside Range comparator mode, either comparator pair A and B or comparator pair C and D can be configured for range comparisons. A single match condition on either of the comparators is recognized as valid. An aligned word access which straddles the range boundary will cause a trigger only if the aligned address is outside the range.

Outside range mode in combination with tagged triggers can be used to detect if the opcode fetches are from an unexpected range. In forced trigger modes the outside range trigger would typically be activated at any interrupt vector fetch or register access. This can be avoided by setting the upper or lower range limit to $7FFFFF or $000000 respectively. Interrupt vector fetches do not cause taghits

When comparing the XGATE address bus in outside range mode, the initial vector fetch as determined by the vector contained in the XGATE XGVBR register should be taken into consideration. The XGVBR register and hence vector address can be modified.

## 8.4.3 Trigger Modes

Trigger modes are used as qualifiers for a state sequencer change of state. The control logic determines the trigger mode and provides a trigger to the state sequencer. The individual trigger modes are described in the following sections.

### 8.4.3.1 Forced Trigger On Comparator Match

If a forced trigger comparator match occurs, the trigger immediately initiates a transition to the next state sequencer state whereby the corresponding flags in DBGSR are set. The state control register for the current state determines the next state for each trigger. Forced triggers are generated as soon as the matching address appears on the address bus, which in the case of opcode fetches occurs several cycles before the opcode execution. For this reason a forced trigger at an opcode address precedes a tagged trigger at the same address by several cycles.

### 8.4.3.2 Trigger On Comparator Related Taghit

If a CPU12X or XGATE taghit occurs, a transition to another state sequencer state is initiated and the corresponding DBGSR flags are set. For a comparator related taghit to occur, the S12XDBG must first generate tags based on comparator matches. When the tagged instruction reaches the execution stage of the instruction queue a taghit is generated by the CPU12X/XGATE. The state control register for the current state determines the next state for each trigger.

### 8.4.3.3 External Tagging Trigger

The $\overline{\text{TAGLO}}$ and $\overline{\text{TAGHI}}$ pins (mapped to device pins) can be used to tag an instruction. This function can be used as another breakpoint source. When the tagged opcode reaches the execution stage of the instruction queue a transition to the disarmed state0 occurs, ending the debug session and generating a breakpoint, if breakpoints are enabled. External tagging is only possible in device emulation modes.

### 8.4.3.4 Trigger On XGATE S/W Breakpoint Request

The XGATE S/W breakpoint request issues a forced breakpoint request to the CPU12X immediately and triggers the state sequencer into the disarmed state. Active tracing sessions are terminated immediately, thus if tracing has not yet begun, no trace information is stored. XGATE generated breakpoints are independent of the DBGBRK bits. The XGSBPE bit in DBGC1 determines if the XGATE S/W breakpoint function is enabled. The BDM bit in DBGC1 determines if the XGATE requested breakpoint causes the system to enter BDM Mode or initiate a software interrupt (SWI).

### 8.4.3.5 TRIG Immediate Trigger

Independent of comparator matches or external tag signals it is possible to initiate a tracing session and/or breakpoint by writing the TRIG bit in DBGC1 to a logic “1”. If configured for begin or mid aligned tracing,

this triggers the state sequencer into the Final State, if configured for end alignment, setting the TRIG bit disarms the module, ending the session. If breakpoints are enabled, a forced breakpoint request is issued immediately (end alignment) or when tracing has completed (begin or mid alignment).

### 8.4.3.6 Trigger Priorities

In case of simultaneous triggers, the priority is resolved according to Table 8-42. The lower priority trigger is suppressed. It is thus possible to miss a lower priority trigger if it occurs simultaneously with a trigger of a higher priority. The trigger priorities described in Table 8-42 dictate that in the case of simultaneous matches, the match on the lower channel number (0,1,2,3) has priority. The SC[3:0] encoding ensures that a match leading to final state has priority over all other matches in each state sequencer state. When configured for range modes a simultaneous match of comparators A and C generates an active match0 whilst match2 is suppressed.

If a write access to DBGC1 with the ARM bit position set occurs simultaneously to a hardware disarm from an internal trigger event, then the ARM bit is cleared due to the hardware disarm.

**Table 8-42. Trigger Priorities**

| Priority | Source | Action |
|---|---|---|
| **Highest** | **XGATE BKP** | Immediate forced breakpoint.....(Tracing terminated immediately). |
| | **TRIG** | Trigger immediately to final state (begin or mid aligned tracing enabled)<br>Trigger immediately to state 0 (end aligned or no tracing enabled) |
| | **External TAGHI/TAGLO** | Enter State0 |
| | **Match0 (force or tag hit)** | Trigger to next state as defined by state control registers |
| | **Match1 (force or tag hit)** | Trigger to next state as defined by state control registers |
| | **Match2 (force or tag hit)** | Trigger to next state as defined by state control registers |
| **Lowest** | **Match3 (force or tag hit)** | Trigger to next state as defined by state control registers |

## 8.4.4 State Sequence Control

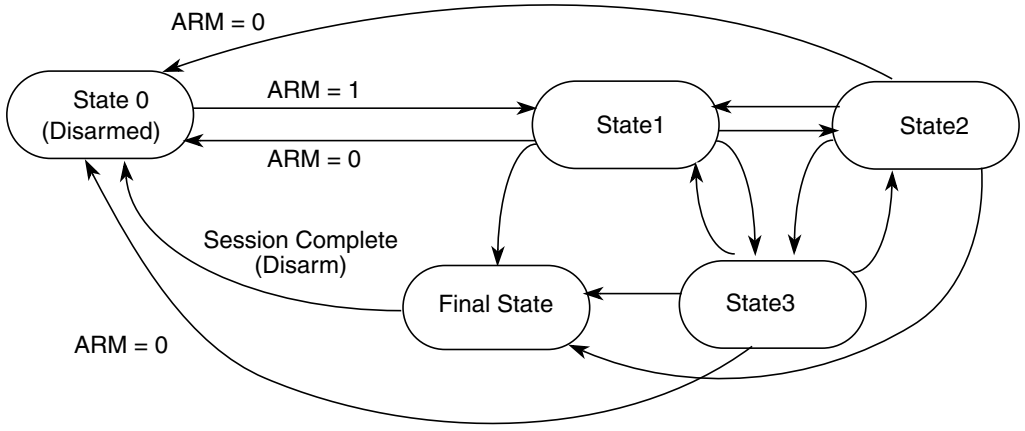



**Figure 8-22. State Sequencer Diagram**

The state sequencer allows a defined sequence of events to provide a trigger point for tracing of data in the trace buffer. Once the S12XDBG module has been armed by setting the ARM bit in the DBGC1 register,

then state1 of the state sequencer is entered. Further transitions between the states are then controlled by the state control registers and depend upon a selected trigger mode condition being met. From Final State the only permitted transition is back to the disarmed state0. Transition between any of the states 1 to 3 is not restricted. Each transition updates the SSF[2:0] flags in DBGSR accordingly to indicate the current state.

Alternatively by setting the TRIG bit in DBGSC1, the state machine can be triggered to state0 or Final State depending on tracing alignment.

A tag hit through $\overline{\text{TAGHI}}$/$\overline{\text{TAGLO}}$ brings the state sequencer immediately into state0, causes a breakpoint, if breakpoints are enabled, and ends tracing immediately independent of the trigger alignment bits TALIGN[1:0].

Independent of the state sequencer, each comparator channel can be individually configured to generate an immediate breakpoint when a match occurs through the use of the BRK bits in the DBGxCTL registers. Thus it is possible to generate an immediate breakpoint on selected channels, whilst a state sequencer transition can be initiated by a match on other channels. If a debug session is ended by a trigger on a channel with BRK = 1, the state sequencer transitions through Final State for a clock cycle to state0. This is independent of tracing and breakpoint activity, thus with tracing and breakpoints disabled, the state sequencer enters state0 and the debug module is disarmed.

An XGATE S/W breakpoint request, if enabled causes a transition to the State0 and generates a breakpoint request to the CPU12X immediately

#### 8.4.4.1 Final State

On entering Final State a trigger may be issued to the trace buffer according to the trace position control as defined by the TALIGN field (see Section 8.3.2.3). If TSOURCE in the trace control register DBGTCR are cleared then the trace buffer is disabled and the transition to Final State can only generate a breakpoint request. In this case or upon completion of a tracing session when tracing is enabled, the ARM bit in the DBGC1 register is cleared, returning the module to the disarmed state0. If tracing is enabled, a breakpoint request can occur at the end of the tracing session. If neither tracing nor breakpoints are enabled then when the final state is reached it returns automatically to state0 and the debug module is disarmed.

### 8.4.5 Trace Buffer Operation

The trace buffer is a 64 lines deep by 64-bits wide RAM array. The S12XDBG module stores trace information in the RAM array in a circular buffer format. The RAM array can be accessed through a register window (DBGTBH:DBGTBL) using 16-bit wide word accesses. After each complete 64-bit trace buffer line is read, an internal pointer into the RAM is incremented so that the next read will receive fresh information. Data is stored in the format shown in Table 8-43. After each store the counter register bits DBGCNT[6:0] are incremented. Tracing of CPU12X activity is disabled when the BDM is active but tracing of XGATE activity is still possible. Reading the trace buffer whilst the DBG is armed returns invalid data and the trace buffer pointer is not incremented.

#### 8.4.5.1 Trace Trigger Alignment

Using the TALIGN bits (see Section 8.3.2.3) it is possible to align the trigger with the end, the middle, or the beginning of a tracing session.

If End or Mid tracing is selected, tracing begins when the ARM bit in DBGC1 is set and State1 is entered. The transition to Final State if End is selected signals the end of the tracing session. The transition to Final State if Mid is selected signals that another 32 lines will be traced before ending the tracing session. Tracing with Begin-Trigger starts at the opcode of the trigger.

##### 8.4.5.1.1 Storing with Begin-Trigger

Storing with Begin-Trigger, data is not stored in the Trace Buffer until the Final State is entered. Once the trigger condition is met the S12XDBG module will remain armed until 64 lines are stored in the Trace Buffer. If the trigger is at the address of the change-of-flow instruction the change of flow associated with the trigger will be stored in the Trace Buffer. Using Begin-trigger together with tagging, if the tagged instruction is about to be executed then the trace is started. Upon completion of the tracing session the breakpoint is generated, thus the breakpoint does not occur at the tagged instruction boundary.

##### 8.4.5.1.2 Storing with Mid-Trigger

Storing with Mid-Trigger, data is stored in the Trace Buffer as soon as the S12XDBG module is armed. When the trigger condition is met, another 32 lines will be traced before ending the tracing session, irrespective of the number of lines stored before the trigger occurred, then the S12XDBG module is disarmed and no more data is stored. Using Mid-trigger with tagging, if the tagged instruction is about to be executed then the trace is continued for another 32 lines. Upon tracing completion the breakpoint is generated, thus the breakpoint does not occur at the tagged instruction boundary.

##### 8.4.5.1.3 Storing with End-Trigger

Storing with End-Trigger, data is stored in the Trace Buffer until the Final State is entered, at which point the S12XDBG module will become disarmed and no more data will be stored. If the trigger is at the address of a change of flow instruction the trigger event will not be stored in the Trace Buffer.

#### 8.4.5.2 Trace Modes

The S12XDBG module can operate in four trace modes. The mode is selected using the TRCMOD bits in the DBGTCR register. In each mode tracing of XGATE or CPU12X information is possible. The source for the trace is selected using the TSOURCE bits in the DBGTCR register. The modes are described in the following subsections. The trace buffer organization is shown in Table 8-43.

##### 8.4.5.2.1 Normal Mode

In Normal Mode, change of flow (COF) program counter (PC) addresses will be stored.

COF addresses are defined as follows for the CPU12X:

- Source address of taken conditional branches (long, short, bit-conditional, and loop primitives)
- Destination address of indexed JMP, JSR, and CALL instruction

- Destination address of RTI, RTS, and RTC instructions.
- Vector address of interrupts, except for SWI and BDM vectors

LBRA, BRA, BSR, BGND as well as non-indexed JMP, JSR, and CALL instructions are not classified as change of flow and are not stored in the trace buffer.

COF addresses are defined as follows for the XGATE:

- Source address of taken conditional branches
- Destination address of indexed JAL instructions.
- First XGATE code address in a thread

Change-of-flow addresses stored include the full 23-bit address bus of CPU12X, the 16-bit address bus for the XGATE module and an information byte, which contains a source/destination bit to indicate whether the stored address was a source address or destination address.

**NOTE**

When an CPU12X COF instruction with destination address is executed, the destination address is stored to the trace buffer on instruction completion, indicating the COF has taken place. If an interrupt occurs simultaneously then the next instruction carried out is actually from the interrupt service routine. The instruction at the destination address of the original program flow gets exectuted after the interrupt service routine.

In the following example an IRQ interrupt occurs during execution of the indexed JMP at address MARK1. The BRN at the destination (SUB_1) is not executed until after the IRQ service routine but the destination address is entered into the trace buffer to indicate that the indexed JMP COF has taken place.

```
         LDX     #SUB_1
MARK1    JMP     0,X                    ; IRQ interrupt occurs during execution of this
MARK2    NOP                            ;

SUB_1    BRN     *                      ; JMP Destination address TRACE BUFFER ENTRY 1
                                        ; RTI Destination address TRACE BUFFER ENTRY 3
         NOP                            ;
ADDR1    DBNE    A,PART5                ; Source address TRACE BUFFER ENTRY 4

IRQ_ISR  LDAB    #$F0                   ; IRQ Vector $FFF2 = TRACE BUFFER ENTRY 2
         STAB    VAR_C1
         RTI                            ;
```

The execution flow taking into account the IRQ is as follows

```
         LDX     #SUB_1
MARK1    JMP     0,X                    ;
IRQ_ISR  LDAB    #$F0                   ;
         STAB    VAR_C1
         RTI                            ;
SUB_1    BRN     *
         NOP                            ;
ADDR1    DBNE    A,PART5                ;
```

#### 8.4.5.2.2 Loop1 Mode

Loop1 Mode, similarly to Normal Mode also stores only COF address information to the trace buffer, it however allows the filtering out of redundant information.

The intent of Loop1 Mode is to prevent the Trace Buffer from being filled entirely with duplicate information from a looping construct such as delays using the DBNE instruction or polling loops using BRSET/BRCLR instructions. Immediately after address information is placed in the Trace Buffer, the S12XDBG module writes this value into a background register. This prevents consecutive duplicate address entries in the Trace Buffer resulting from repeated branches.

Loop1 Mode only inhibits consecutive duplicate source address entries that would typically be stored in most tight looping constructs. It does not inhibit repeated entries of destination addresses or vector addresses, since repeated entries of these would most likely indicate a bug in the user's code that the S12XDBG module is designed to help find.

#### 8.4.5.2.3 Detail Mode

In Detail Mode, address and data for all memory and register accesses is stored in the trace buffer. In the case of XGATE tracing this means that initialization of the R1 register during a vector fetch is not traced. This mode also features information byte entries to the trace buffer, for each address byte entry. The information byte indicates the size of access (word or byte) and the type of access (read or write).

When tracing CPU12X activity in Detail Mode, all cycles are traced except those when the CPU12X is either in a free or opcode fetch cycle. In this mode the XGATE program counter is also traced to provide a snapshot of the XGATE activity. CXINF information byte bits indicate the type of XGATE activity occurring at the time of the trace buffer entry. When tracing CPU12X activity alone in Detail Mode, the address range can be limited to a range specified by the TRANGE bits in DBGTCR. This function uses comparators C and D to define an address range inside which CPU12X activity should be traced (see Table 8-43). Thus the traced CPU12X activity can be restricted to particular register range accesses.

When tracing XGATE activity in Detail Mode, all load and store cycles are traced. Additionally the CPU12X program counter is stored at the time of the XGATE trace buffer entry to provide a snapshot of CPU12X activity.

#### 8.4.5.2.4 Pure PC Mode

In Pure PC Mode, tracing from the CPU the PC addresses of all executed opcodes, including illegal opcodes, are stored. In Pure PC Mode, tracing from the XGATE the PC addresses of all executed opcodes are stored.

### 8.4.5.3 Trace Buffer Organization

Referring to Table 8-43. An X prefix denotes information from the XGATE module, a C prefix denotes information from the CPU12X. ADRH, ADRM, ADRL denote address high, middle and low byte respectively. INF bytes contain control information (R/W, S/D etc.). The numerical suffix indicates which tracing step. The information format for Loop1 Mode and PurePC Mode is the same as that of Normal Mode. Whilst tracing from XGATE or CPU12X only, in Normal or Loop1 modes each array line contains

2 data entries, thus in this case the DBGCNT[0] is incremented after each separate entry. In Detail mode DBGCNT[0] remains cleared whilst the other DBGCNT bits are incremented on each trace buffer entry.

XGATE and CPU12X COFs occur independently of each other and the profile of COFs for the two sources is totally different. When both sources are being traced in Normal or Loop1 mode, for each COF from one source, there may be many COFs from the other source, depending on user code. COF events could occur far from each other in the time domain, on consecutive cycles or simultaneously. When a COF occurs in either source (S12X or XGATE) a trace buffer entry is made and the corresponding CDV or XDV bit is set. The current PC of the other source is simultaneously stored to the trace buffer even if no COF has occurred, in which case CDV/XDV remains cleared indicating the address is not associated with a COF, but is simply a snapshot of the PC contents at the time of the COF from the other source.

Single byte data accesses in Detail Mode are always stored to the low byte of the trace buffer (CDATAL or XDATAL) and the high byte is cleared. When tracing word accesses, the byte at the lower address is always stored to trace buffer byte3 and the byte at the higher address is stored to byte2.

**Table 8-43. Trace Buffer Organization**

| Mode | 8-Byte Wide Word Buffer | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| XGATE Detail | CXINF1 | CADRH1 | CADRM1 | CADRL1 | XDATAH1 | XDATAL1 | XADRM1 | XADRL1 |
| | CXINF2 | CADRH2 | CADRM2 | CADRL2 | XDATAH2 | XDATAL2 | XADRM2 | XADRL2 |
| CPU12X Detail | CXINF1 | CADRH1 | CADRM1 | CADRL1 | CDATAH1 | CDATAL1 | XADRM1 | XADRL1 |
| | CXINF2 | CADRH2 | CADRM2 | CADRL2 | CDATAH2 | CDATAL2 | XADRM2 | XADRL2 |
| Both Other Modes | XINF0 | | XPCM0 | XPCL0 | CINF0 | CPCH0 | CPCM0 | CPCL0 |
| | XINF1 | | XPCM1 | XPCL1 | CINF1 | CPCH1 | CPCM1 | CPCL1 |
| XGATE Other Modes | XINF1 | | XPCM1 | XPCL1 | XINF0 | | XPCM0 | XPCL0 |
| | XINF3 | | XPCM3 | XPCL3 | XINF2 | | XPCM2 | XPCL2 |
| CPU12X Other Modes | CINF1 | CPCH1 | CPCM1 | CPCL1 | CINF0 | CPCH0 | CPCM0 | CPCL0 |
| | CINF3 | CPCH3 | CPCM3 | CPCL3 | CINF2 | CPCH2 | CPCM2 | CPCL2 |

#### 8.4.5.3.1 Information Byte Organization

The format of the control information byte is dependent upon the active trace mode as described below. In Normal, Loop1, or Pure PC modes tracing of XGATE activity, XINF is used to store control information. In Normal, Loop1, or Pure PC modes tracing of CPU12X activity, CINF is used to store control information. In Detail Mode, CXINF contains the control information.

### XGATE Information Byte

| Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|
| XSD | XSOT | XCOT | XDV | 0 | 0 | 0 | 0 |

**Figure 8-23. XGATE Information Byte XINF**

**Table 8-44. XINF Field Descriptions**

| Field | Description |
|---|---|
| 7<br>XSD | **Source Destination Indicator** — This bit indicates if the corresponding stored address is a source or destination address. This is only used in Normal and Loop1 mode tracing.<br>0 Source address<br>1 Destination address or Start of Thread or Continuation of Thread |
| 6<br>XSOT | **Start Of Thread Indicator** — This bit indicates that the corresponding stored address is a start of thread address. This is only used in Normal and Loop1 mode tracing.<br>**NOTE. This bit only has effect on devices where the XGATE module supports multiple interrupt levels**.<br>0 Stored address not from a start of thread<br>1 Stored address from a start of thread |
| 5<br>XCOT | **Continuation Of Thread Indicator** — This bit indicates that the corresponding stored address is the first address following a return from a higher priority thread. This is only used in Normal and Loop1 mode tracing.<br>**NOTE. This bit only has effect on devices where the XGATE module supports multiple interrupt levels.**<br>0 Stored address not from a continuation of thread<br>1 Stored address from a continuation of thread |
| 4<br>XDV | **Data Invalid Indicator** — This bit indicates if the trace buffer entry is invalid. It is only used when tracing from both sources in Normal, Loop1 and Pure PC modes, to indicate that the XGATE trace buffer entry is valid.<br>0 Trace buffer entry is invalid<br>1 Trace buffer entry is valid |

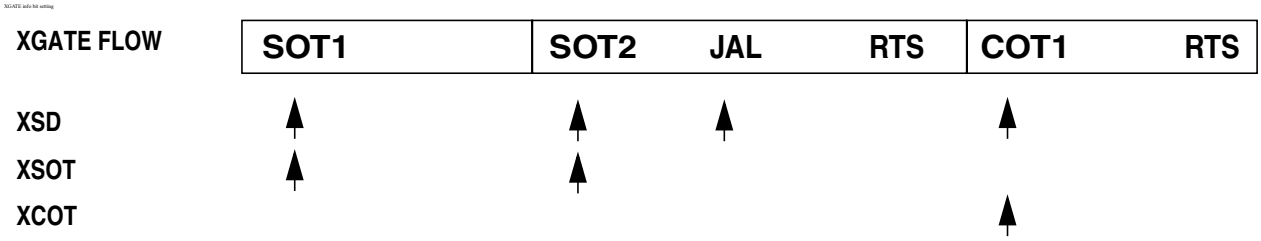



**Figure 8-24. XGATE info bit setting**

Figure 8-24 indicates the XGATE information bit setting when switching between threads, the initial thread starting at SOT1 and continuing at COT1 after the higher priority thread2 has ended.

## CPU12X Information Byte

| Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|
| CSD | CVA | 0 | CDV | 0 | 0 | 0 | 0 |

**Figure 8-25. CPU12X Information Byte CINF**

**Table 8-45. CINF Field Descriptions**

| Field | Description |
|---|---|
| 7<br>CSD | **Source Destination Indicator** — This bit indicates if the corresponding stored address is a source or destination address. This is only used in Normal and Loop1 mode tracing.<br>0 Source address<br>1 Destination address |
| 6<br>CVA | **Vector Indicator** — This bit indicates if the corresponding stored address is a vector address. Vector addresses are destination addresses, thus if CVA is set, then the corresponding CSD is also set. This is only used in Normal and Loop1 mode tracing. This bit has no meaning in Pure PC mode.<br>0 Indexed jump destination address<br>1 Vector destination address |
| 4<br>CDV | **Data Invalid Indicator** — This bit indicates if the trace buffer entry is invalid. It is only used when tracing from both sources in Normal, Loop1 and Pure PC modes, to indicate that the CPU12X trace buffer entry is valid.<br>0 Trace buffer entry is invalid<br>1 Trace buffer entry is valid |

## CXINF Information Byte

| Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|
| CFREE | CSZ | CRW | COCF | XACK | XSZ | XRW | XOCF |

**Figure 8-26. Information Byte CXINF**

This describes the format of the information byte used only when tracing in Detail Mode. When tracing from the CPU12X in Detail Mode, information is stored to the trace buffer on all cycles except opcode fetch and free cycles. The XGATE entry stored on the same line is a snapshot of the XGATE program counter. In this case the CSZ and CRW bits indicate the type of access being made by the CPU12X, whilst the XACK and XOCF bits indicate if the simultaneous XGATE cycle is a free cycle (no bus acknowledge) or opcode fetch cycle. Similarly when tracing from the XGATE in Detail Mode, information is stored to the trace buffer on all cycles except opcode fetch and free cycles. The CPU12X entry stored on the same line is a snapshot of the CPU12X program counter. In this case the XSZ and XRW bits indicate the type of access being made by the XGATE, whilst the CFREE and COCF bits indicate if the simultaneous CPU12X cycle is a free cycle or opcode fetch cycle.

**Table 8-46. CXINF Field Descriptions**

| Field | Description |
|---|---|
| 7<br>CFREE | **CPU12X Free Cycle Indicator** — This bit indicates if the stored CPU12X address corresponds to a free cycle. This bit only contains valid information when tracing the XGATE accesses in Detail Mode.<br>0 Stored information corresponds to free cycle<br>1 Stored information does not correspond to free cycle |

**Table 8-46. CXINF Field Descriptions (continued)**

| Field | Description |
|---|---|
| 6<br>CSZ | **Access Type Indicator** — This bit indicates if the access was a byte or word size access.This bit only contains valid information when tracing CPU12X activity in Detail Mode.<br>0 Word Access<br>1 Byte Access |
| 5<br>CRW | **Read Write Indicator** — This bit indicates if the corresponding stored address corresponds to a read or write access. This bit only contains valid information when tracing CPU12X activity in Detail Mode.<br>0 Write Access<br>1 Read Access |
| 4<br>COCF | **CPU12X Opcode Fetch Indicator** — This bit indicates if the stored address corresponds to an opcode fetch cycle. This bit only contains valid information when tracing the XGATE accesses in Detail Mode.<br>0 Stored information does not correspond to opcode fetch cycle<br>1 Stored information corresponds to opcode fetch cycle |
| 3<br>XACK | **XGATE Access Indicator** — This bit indicates if the stored XGATE address corresponds to a free cycle. This bit only contains valid information when tracing the CPU12X accesses in Detail Mode.<br>0 Stored information corresponds to free cycle<br>1 Stored information does not correspond to free cycle |
| 2<br>XSZ | **Access Type Indicator** — This bit indicates if the access was a byte or word size access. This bit only contains valid information when tracing XGATE activity in Detail Mode.<br>0 Word Access<br>1 Byte Access |
| 1<br>XRW | **Read Write Indicator** — This bit indicates if the corresponding stored address corresponds to a read or write access. This bit only contains valid information when tracing XGATE activity in Detail Mode.<br>0 Write Access<br>1 Read Access |
| 0<br>XOCF | **XGATE Opcode Fetch Indicator** — This bit indicates if the stored address corresponds to an opcode fetch cycle.This bit only contains valid information when tracing the CPU12X accesses in Detail Mode.<br>0 Stored information does not correspond to opcode fetch cycle<br>1 Stored information corresponds to opcode fetch cycle |

### 8.4.5.4 Reading Data from Trace Buffer

The data stored in the Trace Buffer can be read using either the background debug module (BDM) module, the XGATE or the CPU12X provided the S12XDBG module is not armed, is configured for tracing and the system not secured. When the ARM bit is written to 1 the trace buffer is locked to prevent reading. The trace buffer can only be unlocked for reading by an aligned word write to DBGTB when the module is disarmed.

The Trace Buffer can only be read through the DBGTB register using aligned word reads, any byte or misaligned reads return 0 and do not cause the trace buffer pointer to increment to the next trace buffer address. The Trace Buffer data is read out first-in first-out. By reading CNT in DBGCNT the number of valid 64-bit lines can be determined. DBGCNT will not decrement as data is read.

Whilst reading an internal pointer is used to determine the next line to be read. After a tracing session, the pointer points to the oldest data entry, thus if no overflow has occurred, the pointer points to line0, otherwise it points to the line with the oldest entry. The pointer is initialized by each aligned write to DBGTBH to point to the oldest data again. This enables an interrupted trace buffer read sequence to be easily restarted from the oldest data entry.

The least significant word of each 64-bit wide array line is read out first. This corresponds to the bytes 1 and 0 of Table 8-43. The bytes containing invalid information (shaded in Table 8-43) are also read out.

Reading the Trace Buffer while the S12XDBG module is armed will return invalid data and no shifting of the RAM pointer will occur.

#### 8.4.5.5 Trace Buffer Reset State

The Trace Buffer contents are not initialized by a system reset. Thus should a system reset occur, the trace session information from immediately before the reset occurred can be read out. The DBGCNT bits are not cleared by a system reset. Thus should a reset occur, the number of valid lines in the trace buffer is indicated by DBGCNT. The internal pointer to the current trace buffer address is initialized by unlocking the trace buffer thus points to the oldest valid data even if a reset occurred during the tracing session. Generally debugging occurrences of system resets is best handled using mid or end trigger alignment since the reset may occur before the trace trigger, which in the begin trigger alignment case means no information would be stored in the trace buffer.

**NOTE**

An external pin RESET that occurs simultaneous to a trace buffer entry can, in very seldom cases, lead to either that entry being corrupted or the first entry of the session being corrupted. In such cases the other contents of the trace buffer still contain valid tracing information. The case occurs when the reset assertion coincides with the trace buffer entry clock edge.

### 8.4.6 Tagging

A tag follows program information as it advances through the instruction queue. When a tagged instruction reaches the head of the queue a tag hit occurs and triggers the state sequencer.

Each comparator control register features a TAG bit, which controls whether the comparator match will cause a trigger immediately or tag the opcode at the matched address. If a comparator is enabled for tagged comparisons, the address stored in the comparator match address registers must be an opcode address for the trigger to occur.

Both CPU12X and XGATE opcodes can be tagged with the comparator register TAG bits.

Using Begin trigger together with tagging, if the tagged instruction is about to be executed then the transition to the next state sequencer state occurs. If the transition is to the Final State, tracing is started. Only upon completion of the tracing session can a breakpoint be generated. Similarly using Mid trigger with tagging, if the tagged instruction is about to be executed then the trace is continued for another 32 lines. Upon tracing completion the breakpoint is generated. Using End trigger, when the tagged instruction is about to be executed and the next transition is to Final State then a breakpoint is generated immediately, before the tagged instruction is carried out.

Read/Write (R/W), access size (SZ) monitoring and data bus monitoring is not useful if tagged triggering is selected, since the tag is attached to the opcode at the matched address and is not dependent on the data bus nor on the type of access. Thus these bits are ignored if tagged triggering is selected.

When configured for range comparisons and tagging, the ranges are accurate only to word boundaries.

S12X tagging is disabled when the BDM becomes active. XGATE tagging is possible when the BDM is active.

#### 8.4.6.1 External Tagging using $\overline{\text{TAGHI}}$ and $\overline{\text{TAGLO}}$

External tagging using the external $\overline{\text{TAGHI}}$ and $\overline{\text{TAGLO}}$ pins can only be used to tag CPU12X opcodes; tagging of XGATE code using these pins is not possible. An external tag triggers the state sequencer into state0 when the tagged opcode reaches the execution stage of the instruction queue.

The pins operate independently, thus the state of one pin does not affect the function of the other. External tagging is possible in emulation modes only. The presence of logic level 0 on either pin at the rising edge of the external clock (ECLK) performs the function indicated in the Table 8-47. It is possible to tag both bytes of an instruction word. If a taghit occurs, a breakpoint can be generated as defined by the DBGBRK and BDM bits in DBGC1. Each time $\overline{\text{TAGHI}}$ or $\overline{\text{TAGLO}}$ are low on the rising edge of ECLK, the old tag is replaced by a new one.

**Table 8-47. Tag Pin Function**

| $\overline{\text{TAGHI}}$ | $\overline{\text{TAGLO}}$ | Tag |
|---|---|---|
| 1 | 1 | No tag |
| 1 | 0 | Low byte |
| 0 | 1 | High byte |
| 0 | 0 | Both bytes |

#### 8.4.6.2 Unconditional Tagging Function

In emulation modes a low assertion of PE5/$\overline{\text{TAGLO}}$/MODA in the 7th or 8th bus cycle after reset enables the unconditional tagging function, allowing immediate tagging via $\overline{\text{TAGHI}}$/$\overline{\text{TAGLO}}$ with breakpoint to BDM independent of the ARM, BDM and DBGBRK bits. Conversely these bits are not affected by unconditional tagging. The unconditional tagging function remains enabled until the next reset. This function allows an immediate entry to BDM in emulation modes before user code execution. The $\overline{\text{TAGLO}}$ assertion must be in the 7th or 8th bus cycle following the end of reset, whereby the prior $\overline{\text{RESET}}$ pin assertion lasts the full 192 bus cycles.

### 8.4.7 Breakpoints

Breakpoints can be generated as follows.

- Through XGATE software breakpoint requests.
- From comparator channel triggers to final state.
- Using software to write to the TRIG bit in the DBGC1 register.
- From taghits generated using the external $\overline{\text{TAGHI}}$ and $\overline{\text{TAGLO}}$ pins.

Breakpoints generated by the XGATE module or via the BDM BACKGROUND command have no affect on the CPU12X in STOP or WAIT mode.

#### 8.4.7.1 XGATE Software Breakpoints

The XGATE software breakpoint instruction BRK can request a CPU12X breakpoint, via the S12XDBG module. In this case, if the XGSBPE bit is set, the S12XDBG module immediately generates a forced breakpoint request to the CPU12X, the state sequencer is returned to state0 and tracing, if active, is terminated. If configured for BEGIN trigger and tracing has not yet been triggered from another source, the trace buffer contains no information. Breakpoint requests from the XGATE module do not depend upon the state of the DBGBRK or ARM bits in DBGC1. They depend solely on the state of the XGSBPE and BDM bits. Thus it is not necessary to ARM the DBG module to use XGATE software breakpoints to generate breakpoints in the CPU12X program flow, but it is necessary to set XGSBPE. Furthermore, if a breakpoint to BDM is required, the BDM bit must also be set. When the XGATE requests an CPU12X breakpoint, the XGATE program flow stops by default, independent of the S12XDBG module.

#### 8.4.7.2 Breakpoints From Internal Comparator Channel Final State Triggers

Breakpoints can be generated when internal comparator channels trigger the state sequencer to the Final State. If configured for tagging, then the breakpoint is generated when the tagged opcode reaches the execution stage of the instruction queue.

If a tracing session is selected by TSOURCE, breakpoints are requested when the tracing session has completed, thus if Begin or Mid aligned triggering is selected, the breakpoint is requested only on completion of the subsequent trace (see Table 8-48). If no tracing session is selected, breakpoints are requested immediately.

If the BRK bit is set on the triggering channel, then the breakpoint is generated immediately independent of tracing trigger alignment.

**Table 8-48. Breakpoint Setup For Both XGATE and CPU12X Breakpoints**

| BRK | TALIGN | DBGBRK[n] | Breakpoint Alignment |
|---|---|---|---|
| 0 | 00 | 0 | Fill Trace Buffer until trigger<br>(no breakpoints — keep running) |
| 0 | 00 | 1 | Fill Trace Buffer until trigger, then breakpoint request occurs |
| 0 | 01 | 0 | Start Trace Buffer at trigger<br>(no breakpoints — keep running) |
| 0 | 01 | 1 | Start Trace Buffer at trigger<br>A breakpoint request occurs when Trace Buffer is full |
| 0 | 10 | 0 | Store a further 32 Trace Buffer line entries after trigger<br>(no breakpoints — keep running) |
| 0 | 10 | 1 | Store a further 32 Trace Buffer line entries after trigger<br>Request breakpoint after the 32 further Trace Buffer entries |
| 1 | 00,01,10 | 1 | Terminate tracing and generate breakpoint immediately on trigger |
| 1 | 00,01,10 | 0 | Terminate tracing immediately on trigger |
| x | 11 | x | Reserved |

### 8.4.7.3 Breakpoints Generated Via The TRIG Bit

If a TRIG triggers occur, the Final State is entered. If a tracing session is selected by TSOURCE, breakpoints are requested when the tracing session has completed, thus if Begin or Mid aligned triggering is selected, the breakpoint is requested only on completion of the subsequent trace (see Table 8-48). If no tracing session is selected, breakpoints are requested immediately. TRIG breakpoints are possible even if the S12XDBG module is disarmed.

### 8.4.7.4 Breakpoints Via TAGHI Or TAGLO Pin Taghits

Tagging using the external $\overline{\text{TAGHI}}$/$\overline{\text{TAGLO}}$ pins always ends the session immediately at the tag hit. It is always end aligned, independent of internal channel trigger alignment configuration.

### 8.4.7.5 S12XDBG Breakpoint Priorities

XGATE software breakpoints have the highest priority. Active tracing sessions are terminated immediately.

If a TRIG trigger occurs after Begin or Mid aligned tracing has already been triggered by a comparator instigated transition to Final State, then TRIG no longer has an effect. When the associated tracing session is complete, the breakpoint occurs. Similarly if a TRIG is followed by a subsequent trigger from a comparator channel, it has no effect, since tracing has already started.

If a comparator tag hit occurs simultaneously with an external $\overline{\text{TAGHI}}$/$\overline{\text{TAGLO}}$ hit, the state sequencer enters state0. TAGHI/TAGLO triggers are always end aligned, to end tracing immediately, independent of the tracing trigger alignment bits TALIGN[1:0].

#### 8.4.7.5.1 S12XDBG Breakpoint Priorities And BDM Interfacing

Breakpoint operation is dependent on the state of the S12XBDM module. If the S12XBDM module is active, the CPU12X is executing out of BDM firmware and S12X breakpoints are disabled. In addition, while executing a BDM TRACE command, tagging into BDM is disabled. If BDM is not active, the breakpoint will give priority to BDM requests over SWI requests if the breakpoint coincides with a SWI instruction in the user's code. On returning from BDM, the SWI from user code gets executed.

**Table 8-49. Breakpoint Mapping Summary**

| DBGBRK[1] (DBGC1[3]) | BDM Bit (DBGC1[4]) | BDM Enabled | BDM Active | S12X Breakpoint Mapping |
|---|---|---|---|---|
| 0 | X | X | X | No Breakpoint |
| 1 | 0 | X | 0 | Breakpoint to SWI |
| 1 | 0 | X | 1 | No Breakpoint |
| 1 | 1 | 0 | X | Breakpoint to SWI |
| 1 | 1 | 1 | 0 | Breakpoint to BDM |
| 1 | 1 | 1 | 1 | No Breakpoint |

BDM cannot be entered from a breakpoint unless the ENABLE bit is set in the BDM. If entry to BDM via a BGND instruction is attempted and the ENABLE bit in the BDM is cleared, the CPU12X actually

executes the BDM firmware code. It checks the ENABLE and returns if ENABLE is not set. If not serviced by the monitor then the breakpoint is re-asserted when the BDM returns to normal CPU12X flow.

If the comparator register contents coincide with the SWI/BDM vector address then an SWI in user code and DBG breakpoint could occur simultaneously. The CPU12X ensures that BDM requests have a higher priority than SWI requests. Returning from the BDM/SWI service routine care must be taken to avoid re triggering a breakpoint.

**NOTE**

When program control returns from a tagged breakpoint using an RTI or BDM GO command without program counter modification it will return to the instruction whose tag generated the breakpoint. To avoid re triggering a breakpoint at the same location reconfigure the S12XDBG module in the SWI routine, if configured for an SWI breakpoint, or over the BDM interface by executing a TRACE command before the GO to increment the program flow past the tagged instruction.

An XGATE software breakpoint is forced immediately, the tracing session terminated and the XGATE module execution stops. The user can thus determine if an XGATE breakpoint has occurred by reading out the XGATE program counter over the BDM interface.

# Chapter 9
# Security (S12XE9SECV2)

**Table 9-1. Revision History**

| Revision Number | Revision Date | Sections Affected | Description of Changes |
|---|---|---|---|
| V02.00 | 27 Aug 2004 | | - Reviewed and updated for S12XD architecture |
| V02.01 | 21 Feb 2007 | | - Added S12XE, S12XF and S12XS architectures |
| V02.02 | 19 Apr 2007 | | - Corrected statement about Backdoor key access via BDM on XE, XF, XS |

## 9.1 Introduction

This specification describes the function of the security mechanism in the S12XE chip family (9SEC).

**NOTE**

No security feature is absolutely secure. However, Freescale's strategy is to make reading or copying the FLASH and/or EEPROM difficult for unauthorized users.

### 9.1.1 Features

The user must be reminded that part of the security must lie with the application code. An extreme example would be application code that dumps the contents of the internal memory. This would defeat the purpose of security. At the same time, the user may also wish to put a backdoor in the application program. An example of this is the user downloads a security key through the SCI, which allows access to a programming routine that updates parameters stored in another section of the Flash memory.

The security features of the S12XE chip family (in secure mode) are:

- Protect the content of non-volatile memories (Flash, EEPROM)
- Execution of NVM commands is restricted
- Disable access to internal memory via background debug module (BDM)
- Disable access to internal Flash/EEPROM in expanded modes
- Disable debugging features for the CPU and XGATE

## 9.1.2 Modes of Operation

Table 9-2 gives an overview over availability of security relevant features in unsecure and secure modes.

**Table 9-2. Feature Availability in Unsecure and Secure Modes on S12XE**

| | Unsecure Mode | | | | | | Secure Mode | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | NS | SS | NX | ES | EX | ST | NS | SS | NX | ES | EX | ST |
| Flash Array Access | ✔ | ✔ | ✔(1) | ✔1 | ✔1 | ✔1 | ✔ | ✔ | — | — | — | — |
| EEPROM Array Access | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ | — | — | — | — |
| NVM Commands | ✔(2) | ✔ | ✔2 | ✔2 | ✔2 | ✔ | ✔2 | ✔2 | ✔2 | ✔2 | ✔2 | ✔2 |
| BDM | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ | — | ✔(3) | — | — | — | — |
| DBG Module Trace | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ | — | — | — | — | — | — |
| XGATE Debugging | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ | — | — | — | — | — | — |
| External Bus Interface | — | — | ✔ | ✔ | ✔ | ✔ | — | — | ✔ | ✔ | ✔ | ✔ |
| Internal status visible multiplexed on external bus | — | — | — | ✔ | ✔ | — | — | — | — | ✔ | ✔ | — |
| Internal accesses visible on external bus | — | — | — | — | — | ✔ | — | — | — | — | — | ✔ |

1. Availability of Flash arrays in the memory map depends on ROMCTL/EROMCTL pins and/or the state of the ROMON/EROMON bits in the MMCCTL1 register. Please refer to the S12X_MMC block guide for detailed information.
2. Restricted NVM command set only. Please refer to the NVM wrapper block guides for detailed information.
3. BDM hardware commands restricted to peripheral registers only.

## 9.1.3 Securing the Microcontroller

Once the user has programmed the Flash and EEPROM, the chip can be secured by programming the security bits located in the options/security byte in the Flash memory array. These non-volatile bits will keep the device secured through reset and power-down.

The options/security byte is located at address 0xFF0F (= global address 0x7F_FF0F) in the Flash memory array. This byte can be erased and programmed like any other Flash location. Two bits of this byte are used for security (SEC[1:0]). On devices which have a memory page window, the Flash options/security byte is also available at address 0xBF0F by selecting page 0x3F with the PPAGE register. The contents of this byte are copied into the Flash security register (FSEC) during a reset sequence.

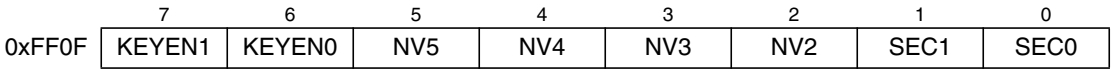



**Figure 9-1. Flash Options/Security Byte**

The meaning of the bits KEYEN[1:0] is shown in Table 9-3. Please refer to Section 9.1.5.1, “Unsecuring the MCU Using the Backdoor Key Access” for more information.

**Table 9-3. Backdoor Key Access Enable Bits**

| KEYEN[1:0] | Backdoor Key Access Enabled |
|---|---|
| 00 | 0 (disabled) |
| 01 | 0 (disabled) |
| 10 | 1 (enabled) |
| 11 | 0 (disabled) |

The meaning of the security bits SEC[1:0] is shown in Table 9-4. For security reasons, the state of device security is controlled by two bits. To put the device in unsecured mode, these bits must be programmed to SEC[1:0] = ‘10’. All other combinations put the device in a secured mode. The recommended value to put the device in secured state is the inverse of the unsecured state, i.e. SEC[1:0] = ‘01’.

**Table 9-4. Security Bits**

| SEC[1:0] | Security State |
|---|---|
| 00 | 1 (secured) |
| 01 | 1 (secured) |
| ***10*** | ***0 (unsecured)*** |
| 11 | 1 (secured) |

**NOTE**

Please refer to the Flash block guide for actual security configuration (in section “Flash Module Security”).

## 9.1.4 Operation of the Secured Microcontroller

By securing the device, unauthorized access to the EEPROM and Flash memory contents can be prevented. However, it must be understood that the security of the EEPROM and Flash memory contents also depends on the design of the application program. For example, if the application has the capability of downloading code through a serial port and then executing that code (e.g. an application containing bootloader code), then this capability could potentially be used to read the EEPROM and Flash memory contents even when the microcontroller is in the secure state. In this example, the security of the application could be enhanced by requiring a challenge/response authentication before any code can be downloaded.

Secured operation has the following effects on the microcontroller:

### 9.1.4.1 Normal Single Chip Mode (NS)

- Background debug module (BDM) operation is completely disabled.
- Execution of Flash and EEPROM commands is restricted. Please refer to the NVM block guide for details.
- Tracing code execution using the DBG module is disabled.
- Debugging XGATE code (breakpoints, single-stepping) is disabled.

#### 9.1.4.2 Special Single Chip Mode (SS)

- BDM firmware commands are disabled.
- BDM hardware commands are restricted to the register space.
- Execution of Flash and EEPROM commands is restricted. Please refer to the NVM block guide for details.
- Tracing code execution using the DBG module is disabled.
- Debugging XGATE code (breakpoints, single-stepping) is disabled.

Special single chip mode means BDM is active after reset. The availability of BDM firmware commands depends on the security state of the device. The BDM secure firmware first performs a blank check of both the Flash memory and the EEPROM. If the blank check succeeds, security will be temporarily turned off and the state of the security bits in the appropriate Flash memory location can be changed If the blank check fails, security will remain active, only the BDM hardware commands will be enabled, and the accessible memory space is restricted to the peripheral register area. This will allow the BDM to be used to erase the EEPROM and Flash memory without giving access to their contents. After erasing both Flash memory and EEPROM, another reset into special single chip mode will cause the blank check to succeed and the options/security byte can be programmed to “unsecured” state via BDM.

While the BDM is executing the blank check, the BDM interface is completely blocked, which means that all BDM commands are temporarily blocked.

#### 9.1.4.3 Expanded Modes (NX, ES, EX, and ST)

- BDM operation is completely disabled.
- Internal Flash memory and EEPROM are disabled.
- Execution of Flash and EEPROM commands is restricted. Please refer to the FTM block guide for details.
- Tracing code execution using the DBG module is disabled.
- Debugging XGATE code (breakpoints, single-stepping) is disabled

### 9.1.5 Unsecuring the Microcontroller

Unsecuring the microcontroller can be done by three different methods:

1. Backdoor key access
2. Reprogramming the security bits
3. Complete memory erase (special modes)

#### 9.1.5.1 Unsecuring the MCU Using the Backdoor Key Access

In normal modes (single chip and expanded), security can be temporarily disabled using the backdoor key access method. This method requires that:

- The backdoor key at 0xFF00–0xFF07 (= global addresses 0x7F_FF00–0x7F_FF07) has been programmed to a valid value.

- The KEYEN[1:0] bits within the Flash options/security byte select ‘enabled’.
- In single chip mode, the application program programmed into the microcontroller must be designed to have the capability to write to the backdoor key locations.

The backdoor key values themselves would not normally be stored within the application data, which means the application program would have to be designed to receive the backdoor key values from an external source (e.g. through a serial port).

The backdoor key access method allows debugging of a secured microcontroller without having to erase the Flash. This is particularly useful for failure analysis.

**NOTE**

No word of the backdoor key is allowed to have the value 0x0000 or 0xFFFF.

### 9.1.6 Reprogramming the Security Bits

In normal single chip mode (NS), security can also be disabled by erasing and reprogramming the security bits within Flash options/security byte to the unsecured value. Because the erase operation will erase the entire sector from 0xFE00–0xFFFF (0x7F_FE00–0x7F_FFFF), the backdoor key and the interrupt vectors will also be erased; this method is not recommended for normal single chip mode. The application software can only erase and program the Flash options/security byte if the Flash sector containing the Flash options/security byte is not protected (see Flash protection). Thus Flash protection is a useful means of preventing this method. The microcontroller will enter the unsecured state after the next reset following the programming of the security bits to the unsecured value.

This method requires that:

- The application software previously programmed into the microcontroller has been designed to have the capability to erase and program the Flash options/security byte, or security is first disabled using the backdoor key method, allowing BDM to be used to issue commands to erase and program the Flash options/security byte.
- The Flash sector containing the Flash options/security byte is not protected.

### 9.1.7 Complete Memory Erase (Special Modes)

The microcontroller can be unsecured in special modes by erasing the entire EEPROM and Flash memory contents.

When a secure microcontroller is reset into special single chip mode (SS), the BDM firmware verifies whether the EEPROM and Flash memory are erased. If any EEPROM or Flash memory address is not erased, only BDM hardware commands are enabled. BDM hardware commands can then be used to write to the EEPROM and Flash registers to mass erase the EEPROM and all Flash memory blocks.

When next reset into special single chip mode, the BDM firmware will again verify whether all EEPROM and Flash memory are erased, and this being the case, will enable all BDM commands, allowing the Flash options/security byte to be programmed to the unsecured value. The security bits SEC[1:0] in the Flash security register will indicate the unsecure state following the next reset.

# Chapter 10
# XGATE (S12XGATEV3)

**Table 10-1. Revision History**

| Revision Number | Revision Date | Sections Affected | Description of Changes |
|---|---|---|---|
| V03.22 | 06 Oct 2005 | | - Internal updates |
| V03.23 | 14 Dec 2005 | 10.9.2/10-463 | - Updated code example |
| V03.24 | 17 Jan 2006 | | - Internal updates |

## 10.1 Introduction

The XGATE module is a peripheral co-processor that allows autonomous data transfers between the MCU's peripherals and the internal memories. It has a built in RISC core that is able to pre-process the transferred data and perform complex communication protocols.

The XGATE module is intended to increase the MCU's data throughput by lowering the S12X_CPU's interrupt load.

Figure 10-1 gives an overview on the XGATE architecture.

This document describes the functionality of the XGATE module, including:

- XGATE registers (Section 10.3, "Memory Map and Register Definition")
- XGATE RISC core (Section 10.4.1, "XGATE RISC Core")
- Hardware semaphores (Section 10.4.4, "Semaphores")
- Interrupt handling (Section 10.5, "Interrupts")
- Debug features (Section 10.6, "Debug Mode")
- Security (Section 10.7, "Security")
- Instruction set (Section 10.8, "Instruction Set")

### 10.1.1 Glossary of Terms

XGATE Request

A service request from a peripheral module which is directed to the XGATE by the S12X_INT module (see Figure 10-1). Each XGATE request attempts to activate a XGATE channel at a certain priority level.

XGATE Channel

The resources in the XGATE module (i.e. Channel ID number, Priority level, Service Request Vector, Interrupt Flag) which are associated with a particular XGATE Request.

XGATE Channel ID

A 7-bit identifier associated with an XGATE channel. In S12XE designs valid Channel IDs range from $0D to $78.

XGATE Priority Level

A priority ranging from 1 to 7 which is associated with an XGATE channel. The priority level of an XGATE channel is selected in the S12X_INT module.

XGATE Register Bank

A register bank consists of registers R1-R7, CCR and the PC. Each interrupt level is associated with one register bank.

XGATE Channel Interrupt

An S12X_CPU interrupt that is triggered by a code sequence running on the XGATE module.

XGATE Software Channel

Special XGATE channel that is not associated with any peripheral service request. A Software Channel is triggered by its Software Trigger Bit which is implemented in the XGATE module.

XGATE Semaphore

A set of hardware flip-flops that can be exclusively set by either the S12X_CPU or the XGATE. (see Section 10.4.4, “Semaphores”)

XGATE Thread

A code sequence which is executed by the XGATE’s RISC core after receiving an XGATE request.

XGATE Debug Mode

A special mode in which the XGATE’s RISC core is halted for debug purposes. This mode enables the XGATE’s debug features (see Section 10.6, “Debug Mode”).

XGATE Software Error

The XGATE is able to detect a number of error conditions caused by erratic software (see Section 10.4.5, “Software Error Detection”). These error conditions will cause the XGATE to seize program execution and flag an Interrupt to the S12X_CPU.

Word

A 16 bit entity.

Byte

An 8 bit entity.

## 10.1.2 Features

The XGATE module includes these features:

- Data movement between various targets (i.e. Flash, RAM, and peripheral modules)
- Data manipulation through built in RISC core

- Provides up to 108 XGATE channels, including 8 software triggered channels
- Interruptible thread execution
- Two register banks to support fast context switching between threads
- Hardware semaphores which are shared between the S12X_CPU and the XGATE module
- Able to trigger S12X_CPU interrupts upon completion of an XGATE transfer
- Software error detection to catch erratic application code

### 10.1.3 Modes of Operation

There are four run modes on S12XE devices.

- Run mode, wait mode, stop mode

  The XGATE is able to operate in all of these three system modes. Clock activity will be automatically stopped when the XGATE module is idle.
- Freeze mode (BDM active)

  In freeze mode all clocks of the XGATE module may be stopped, depending on the module configuration (see Section 10.3.1.1, “XGATE Control Register (XGMCTL)”).

### 10.1.4 Block Diagram

Figure 10-1 shows a block diagram of the XGATE.

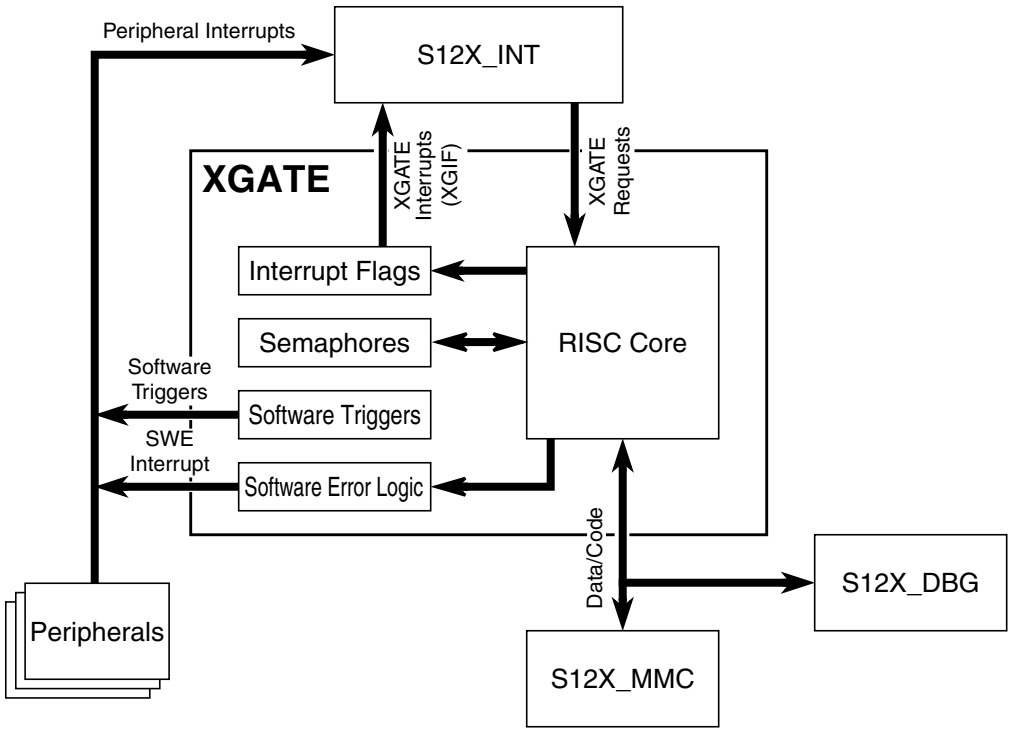



**Figure 10-1. XGATE Block Diagram**

## 10.2 External Signal Description

The XGATE module has no external pins.

## 10.3 Memory Map and Register Definition

This section provides a detailed description of address space and registers used by the XGATE module.

The memory map for the XGATE module is given below in Figure 10-2.The address listed for each register is the sum of a base address and an address offset. The base address is defined at the SoC level and the address offset is defined at the module level. Reserved registers read zero. Write accesses to the reserved registers have no effect.

### 10.3.1 Register Descriptions

This section consists of register descriptions in address order. Each description includes a standard register diagram with an associated figure number. Details of register bits and field functions follow the register diagrams, in bit order.

| Register Name | | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0x0000 XGMCTL | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | XGE | XGFRZ | XGDBG | XGSS | XG FACT | 0 | XG SWEF | XGIE |
| | W | XGEM | XG FRZM | XG DBGM | XGSSM | XG FACTM | | XG SWEFM | XGIEM | | | | | | | | |
| 0x0002 XGCHID | R | | | | | | | | | 0 | XGCHID[6:0] | | | | | | |
| | W | | | | | | | | | | | | | | | | |
| 0x0003 XGCHPL | R | | | | | | | | | 0 | 0 | 0 | 0 | 0 | XGCHPL[2:0] | | |
| | W | | | | | | | | | | | | | | | | |
| 0x0004 Reserved | R | | | | | | | | | | | | | | | | |
| | W | | | | | | | | | | | | | | | | |
| 0x0005 XGISPSEL | R | | | | | | | | | 0 | 0 | 0 | 0 | 0 | 0 | XGISPSEL[1:0] | |
| | W | | | | | | | | | | | | | | | | |
| 0x0006 XGISP74 | R | XGISP74[15:1] | | | | | | | | | | | | | | | 0 |
| | W | | | | | | | | | | | | | | | | |
| 0x0006 XGISP31 | R | XGISP31[15:1] | | | | | | | | | | | | | | | 0 |
| | W | | | | | | | | | | | | | | | | |
| 0x0006 XGVBR | R | XGVBR[15:1] | | | | | | | | | | | | | | | 0 |
| | W | | | | | | | | | | | | | | | | |

= Unimplemented or Reserved

**Figure 10-2. XGATE Register Summary (Sheet 1 of 3)**

| | | 127 | 126 | 125 | 124 | 123 | 122 | 121 | 120 | 119 | 118 | 117 | 116 | 115 | 114 | 113 | 112 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0x0008 XGIF | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | XGIF_78 | XGF_77 | XGIF_76 | XGIF_75 | XGIF_74 | XGIF_73 | XGIF_72 | XGIF_71 | XGIF_70 |
| | W | | | | | | | | | | | | | | | | |

| | | 111 | 110 | 109 | 108 | 107 | 106 | 105 | 104 | 103 | 102 | 101 | 100 | 99 | 98 | 97 | 96 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0x000A XGIF | R | XGIF_6F | XGIF_6E | XGIF_6D | XGIF_6C | XGIF_6B | XGIF_6A | XGIF_69 | XGIF_68 | XGF_67 | XGIF_66 | XGIF_65 | XGIF_64 | XGIF_63 | XGIF_62 | XGIF_61 | XGIF_60 |
| | W | | | | | | | | | | | | | | | | |

| | | 95 | 94 | 93 | 92 | 91 | 90 | 89 | 88 | 87 | 86 | 85 | 84 | 83 | 82 | 81 | 80 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0x000C XGIF | R | XGIF_5F | XGIF_5E | XGIF_5D | XGIF_5C | XGIF_5B | XGIF_5A | XGIF_59 | XGIF_58 | XGF_57 | XGIF_56 | XGIF_55 | XGIF_54 | XGIF_53 | XGIF_52 | XGIF_51 | XGIF_50 |
| | W | | | | | | | | | | | | | | | | |

| | | 79 | 78 | 77 | 76 | 75 | 74 | 73 | 72 | 71 | 70 | 69 | 68 | 67 | 66 | 65 | 64 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0x000E XGIF | R | XGIF_4F | XGIF_4E | XGIF_4D | XGIF_4C | XGIF_4B | XGIF_4A | XGIF_49 | XGIF_48 | XGF _47 | XGIF_46 | XGIF_45 | XGIF_44 | XGIF_43 | XGIF_42 | XGIF_41 | XGIF_40 |
| | W | | | | | | | | | | | | | | | | |

| | | 63 | 62 | 61 | 60 | 59 | 58 | 57 | 56 | 55 | 54 | 53 | 52 | 51 | 50 | 49 | 48 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0x0010 XGIF | R | XGIF_3F | XGIF_3E | XGIF_3D | XGIF_3C | XGIF_3B | XGIF_3A | XGIF_39 | XGIF_38 | XGF _37 | XGIF_36 | XGIF_35 | XGIF_34 | XGIF_33 | XGIF_32 | XGIF_31 | XGIF_30 |
| | W | | | | | | | | | | | | | | | | |

| | | 47 | 46 | 45 | 44 | 43 | 42 | 41 | 40 | 39 | 38 | 37 | 36 | 35 | 34 | 33 | 32 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0x0012 XGIF | R | XGIF_2F | XGIF_2E | XGIF_2D | XGIF_2C | XGIF_2B | XGIF_2A | XGIF_29 | XGIF_28 | XGF _27 | XGIF_26 | XGIF_25 | XGIF_24 | XGIF_23 | XGIF_22 | XGIF_21 | XGIF_20 |
| | W | | | | | | | | | | | | | | | | |

| | | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0x0014 XGIF | R | XGIF_1F | XGIF_1E | XGIF_1D | XGIF_1C | XGIF_1B | XGIF_1A | XGIF_19 | XGIF_18 | XGF _17 | XGIF_16 | XGIF_15 | XGIF_14 | XGIF_13 | XGIF_12 | XGIF_11 | XGIF_10 |
| | W | | | | | | | | | | | | | | | | |

| | | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0x0016 XGIF | R | XGIF_0F | XGIF_0E | XGIF_0D | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | W | | | | | | | | | | | | | | | | |

= Unimplemented or Reserved

**Figure 10-2. XGATE Register Summary (Sheet 2 of 3)**

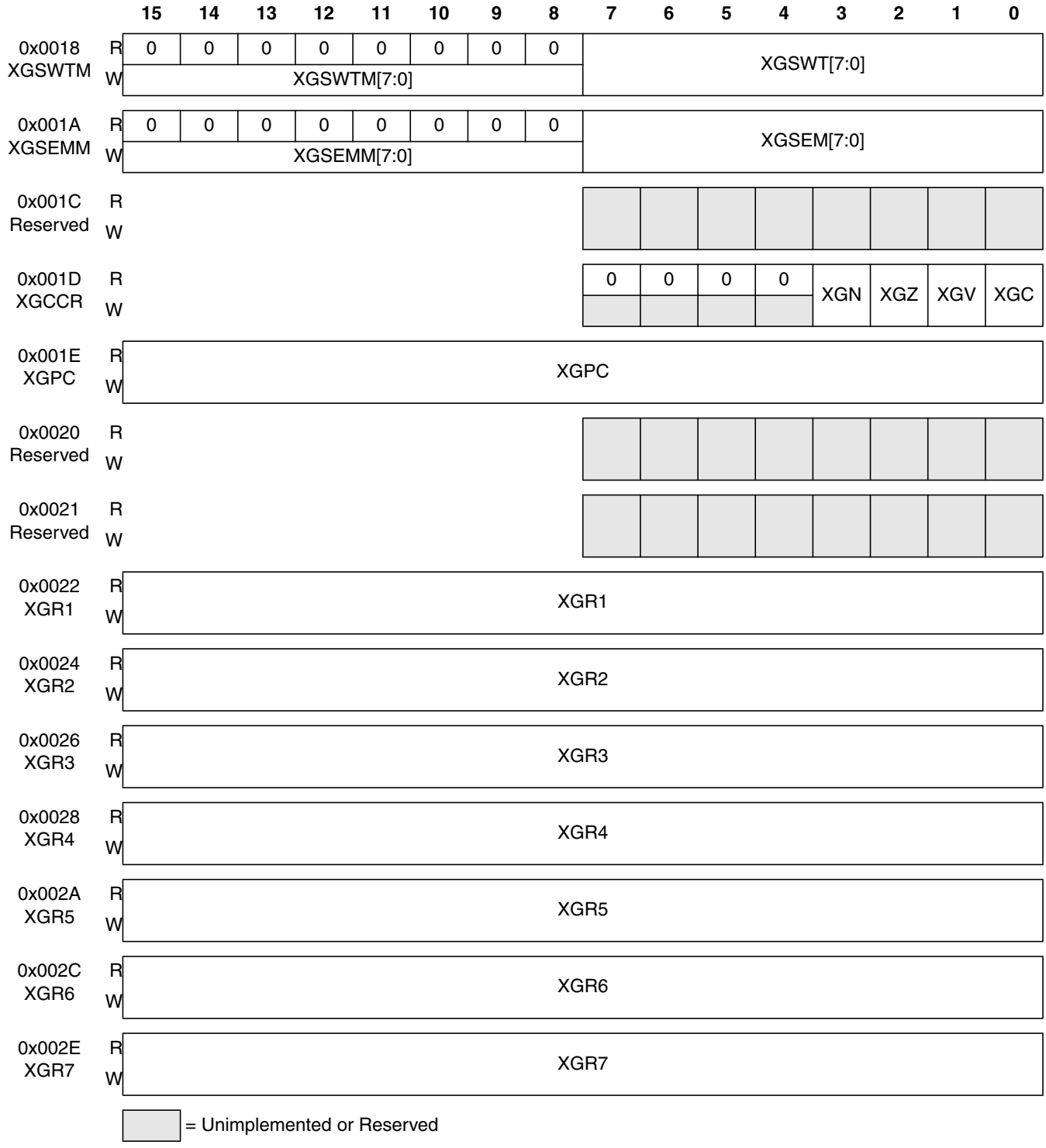



**Figure 10-2. XGATE Register Summary (Sheet 3 of 3)**

### 10.3.1.1 XGATE Control Register (XGMCTL)

All module level switches and flags are located in the XGATE Module Control Register Figure 10-3.

Module Base +0x00000

| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | XGE | XGFRZ | XGDBG | XGSS | XGFACT | 0 | XG SWEF | XGIE |
| W | XGEM | XG FRZM | XG DBGM | XG SSM | XG FACTM | | XG SWEFM | XGIEM | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented or Reserved

**Figure 10-3. XGATE Control Register (XGMCTL)**

Read: Anytime

Write: Anytime

**Table 10-2. XGMCTL Field Descriptions (Sheet 1 of 3)**

| Field | Description |
|---|---|
| 15<br>XGEM | **XGE Mask** — This bit controls the write access to the XGE bit. The XGE bit can only be set or cleared if a "1" is written to the XGEM bit in the same register access.<br>Read:<br>This bit will always read "0".<br>Write:<br>0 Disable write access to the XGE in the same bus cycle<br>1 Enable write access to the XGE in the same bus cycle |
| 14<br>XGFRZM | **XGFRZ Mask** — This bit controls the write access to the XGFRZ bit. The XGFRZ bit can only be set or cleared if a "1" is written to the XGFRZM bit in the same register access.<br>Read:<br>This bit will always read "0".<br>Write:<br>0 Disable write access to the XGFRZ in the same bus cycle<br>1 Enable write access to the XGFRZ in the same bus cycle |
| 13<br>XGDBGM | **XGDBG Mask** — This bit controls the write access to the XGDBG bit. The XGDBG bit can only be set or cleared if a "1" is written to the XGDBGM bit in the same register access.<br>Read:<br>This bit will always read "0".<br>Write:<br>0 Disable write access to the XGDBG in the same bus cycle<br>1 Enable write access to the XGDBG in the same bus cycle |
| 12<br>XGSSM | **XGSS Mask** — This bit controls the write access to the XGSS bit. The XGSS bit can only be set or cleared if a "1" is written to the XGSSM bit in the same register access.<br>Read:<br>This bit will always read "0".<br>Write:<br>0 Disable write access to the XGSS in the same bus cycle<br>1 Enable write access to the XGSS in the same bus cycle |

**Table 10-2. XGMCTL Field Descriptions (Sheet 2 of 3)**

| Field | Description |
|---|---|
| 11<br>XGFACTM | **XGFACT Mask** — This bit controls the write access to the XGFACT bit. The XGFACT bit can only be set or cleared if a "1" is written to the XGFACTM bit in the same register access.<br>Read:<br>This bit will always read "0".<br>Write:<br>0 Disable write access to the XGFACT in the same bus cycle<br>1 Enable write access to the XGFACT in the same bus cycle |
| 9<br>XGSWEFM | **XGSWEF Mask** — This bit controls the write access to the XGSWEF bit. The XGSWEF bit can only be cleared if a "1" is written to the XGSWEFM bit in the same register access.<br>Read:<br>This bit will always read "0".<br>Write:<br>0 Disable write access to the XGSWEF in the same bus cycle<br>1 Enable write access to the XGSWEF in the same bus cycle |
| 8<br>XGIEM | **XGIE Mask** — This bit controls the write access to the XGIE bit. The XGIE bit can only be set or cleared if a "1" is written to the XGIEM bit in the same register access.<br>Read:<br>This bit will always read "0".<br>Write:<br>0 Disable write access to the XGIE in the same bus cycle<br>1 Enable write access to the XGIE in the same bus cycle |
| 7<br>XGE | **XGATE Module Enable (Request Enable)**— This bit enables incoming XGATE requests from the S12X_INT module. If the XGE bit is cleared, pending XGATE requests will be ignored. The thread that is executed by the RISC core while the XGE bit is cleared will continue to run.<br>Read:<br>0 Incoming requests are disabled<br>1 Incoming requests are enabled<br>Write:<br>0 Disable incoming requests<br>1 Enable incoming requests |
| 6<br>XGFRZ | **Halt XGATE in Freeze Mode** — The XGFRZ bit controls the XGATE operation in Freeze Mode (BDM active).<br>Read:<br>0 RISC core operates normally in Freeze (BDM active)<br>1 RISC core stops in Freeze Mode (BDM active)<br>Write:<br>0 Don’t stop RISC core in Freeze Mode (BDM active)<br>1 Stop RISC core in Freeze Mode (BDM active) |
| 5<br>XGDBG | **XGATE Debug Mode** — This bit indicates that the XGATE is in Debug Mode (see Section 10.6, “Debug Mode”). Debug Mode can be entered by Software Breakpoints (BRK instruction), Tagged or Forced Breakpoints (see **S12X_DBG Section**), or by writing a "1" to this bit.<br>Read:<br>0 RISC core is not in Debug Mode<br>1 RISC core is in Debug Mode<br>Write:<br>0 Leave Debug Mode<br>1 Enter Debug Mode<br>**Note:** Freeze Mode and Software Error Interrupts have no effect on the XGDBG bit. |

**Table 10-2. XGMCTL Field Descriptions (Sheet 3 of 3)**

| Field | Description |
|---|---|
| 4<br>XGSS | **XGATE Single Step** — This bit forces the execution of a single instruction.(1)<br>Read:<br>0 No single step in progress<br>1 Single step in progress<br>Write<br>0 No effect<br>1 Execute a single RISC instruction<br>**Note:** Invoking a Single Step will cause the XGATE to temporarily leave Debug Mode until the instruction has been executed. |
| 3<br>XGFACT | **Fake XGATE Activity** — This bit forces the XGATE to flag activity to the MCU even when it is idle. When it is set the MCU will never enter system stop mode which assures that peripheral modules will be clocked during XGATE idle periods<br>Read:<br>0 XGATE will only flag activity if it is not idle or in debug mode.<br>1 XGATE will always signal activity to the MCU.<br>Write:<br>0 Only flag activity if not idle or in debug mode.<br>1 Always signal XGATE activity. |
| 1<br>XGSWEF | **XGATE Software Error Flag** — This bit signals a software error. It is set whenever the RISC core detects an error condition(2). The RISC core is stopped while this bit is set. Clearing this bit will terminate the current thread and cause the XGATE to become idle.<br>Read:<br>0 No software error detected<br>1 Software error detected<br>Write:<br>0 No effect<br>1 Clears the XGSWEF bit |
| 0<br>XGIE | **XGATE Interrupt Enable** — This bit acts as a global interrupt enable for the XGATE module<br>Read:<br>0 All outgoing XGATE interrupts disabled (except software error interrupts)<br>1 All outgoing XGATE interrupts enabled<br>Write:<br>0 Disable all outgoing XGATE interrupts (except software error interrupts)<br>1 Enable all outgoing XGATE interrupts |

1. Refer to Section 10.6.1, “Debug Features”
2. Refer to Section 10.4.5, “Software Error Detection”

### 10.3.1.2 XGATE Channel ID Register (XGCHID)

The XGATE Channel ID Register (Figure 10-4) shows the identifier of the XGATE channel that is currently active. This register will read “$00” if the XGATE module is idle. In debug mode this register can be used to start and terminate threads. Refer to Section 10.6.1, “Debug Features” for further information.

Module Base +0x0002

**Figure 10-4. XGATE Channel ID Register (XGCHID)**

Read: Anytime

Write: In Debug Mode[1]

**Table 10-3. XGCHID Field Descriptions**

| Field | Description |
|---|---|
| 6–0 XGCHID[6:0] | **Request Identifier** — ID of the currently active channel |

### 10.3.1.3 XGATE Channel Priority Level (XGCHPL)

The XGATE Channel Priority Level Register (Figure 10-5) shows the priority level of the current thread. In debug mode this register can be used to select a priority level when launching a thread (see Section 10.6.1, “Debug Features”).

Module Base +0x0003

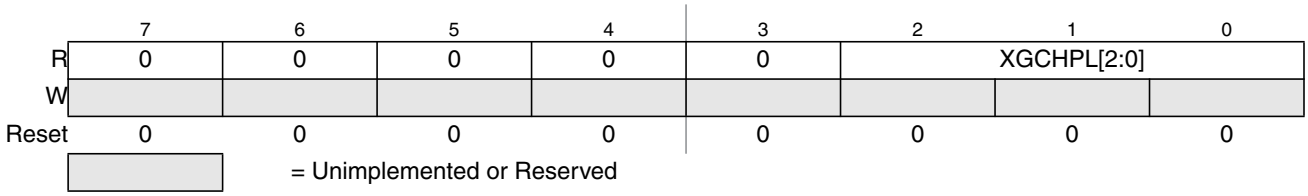



**Figure 10-5. XGATE Channel Priority Level Register (XGCHPL)**

Read: Anytime

Write: In Debug Mode[1]

**Table 10-4. XGCHPL Field Descriptions**

| Field | Description |
|---|---|
| 2-0 XGCHPL[2:0] | **Priority Level**— Priority level of the currently active channel |

### 10.3.1.4 XGATE Initial Stack Pointer Select Register (XGISPSEL)

The XGATE Initial Stack Pointer Select Register (Figure 10-6) determines the register which is mapped to address “Module Base +0x0006”. A value of zero selects the Vector Base Register (XGVBR). Setting

1. Refer to Section 10.6.1, “Debug Features”

this register to a channel priority level (non-zero value) selects the corresponding Initial Stack Pointer Registers XGISP74 or XGISP31 (see Table 10-6).

Module Base +0x0005

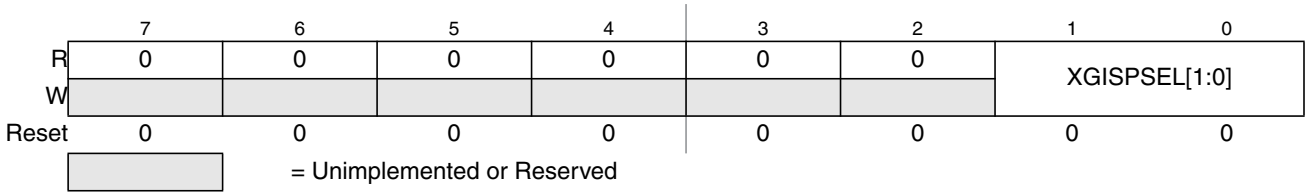



**Figure 10-6. XGATE Initial Stack Pointer Select Register (XGISPSEL)**

Read: Anytime

Write: Anytime

**Table 10-5. XGISPSEL Field Descriptions**

| Field | Description |
|---|---|
| 1-0<br>XGISPSEL[1:0] | **Register select**— Determines whether XGISP74, XGISP31, or XGVBR is mapped to “Module Base +0x0006”. See Table 10-6. |

**Table 10-6. XGISP74, XGISP31, XGVBR Mapping**

| XGISPSEL[1:0] | Register Mapped to “Module Base +0x0006“ |
|---|---|
| 3 | Reserved |
| 2 | XGISP74 |
| 1 | XGISP31 |
| 0 | XGVBR |

### 10.3.1.5 XGATE Initial Stack Pointer for Interrupt Priorities 7 to 4 (XGISP74)

The XGISP74 register is intended to point to the stack region that is used by XGATE channels of priority 7 to 4. Every time a thread of such priority is started, RISC core register R7 will be initialized with the content of XGISP74.

Module Base +0x0006

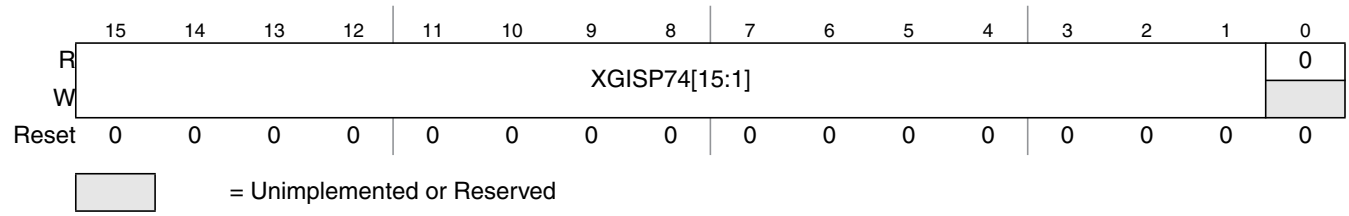



**Figure 10-7. XGATE Initial Stack Pointer for Interrupt Priorities 7 to 4 (XGISP74)**

Read: Anytime

Write: Only if XGATE requests are disabled (XGE = 0) and idle (XGCHID = $00))

**Table 10-7. XGISP74 Field Descriptions**

| Field | Description |
|---|---|
| 15–1<br>XBISP74[15:1] | **Initial Stack Pointer**— The XGISP74 register holds the initial value of RISC core register R7, for threads of priority 7 to 4. |

### 10.3.1.6 XGATE Initial Stack Pointer for Interrupt Priorities 3 to 1 (XGISP31)

The XGISP31 register is intended to point to the stack region that is used by XGATE channels of priority 3 to 1. Every time a thread of such priority is started, RISC core register R7 will be initialized with the content of XGISP31.

Module Base +0x0006

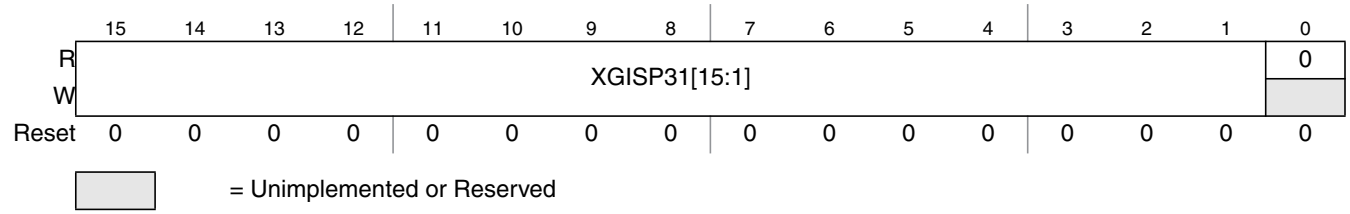



**Figure 10-8. XGATE Initial Stack Pointer for Interrupt Priorities 3 to 1 (XGISP31)**

Read: Anytime

Write: Only if XGATE requests are disabled (XGE = 0) and idle (XGCHID = $00))

**Table 10-8. XGISP31 Field Descriptions**

| Field | Description |
|---|---|
| 15–1<br>XBISP31[15:1] | **Initial Stack Pointer**— The XGISP31 register holds the initial value of RISC core register R7, for threads of priority 3 to 1. |

### 10.3.1.7 XGATE Vector Base Address Register (XGVBR)

The Vector Base Address Register (Figure 10-9) determines the location of the XGATE vector block (see Section Figure 10-23., “XGATE Vector Block).

Module Base +0x0006

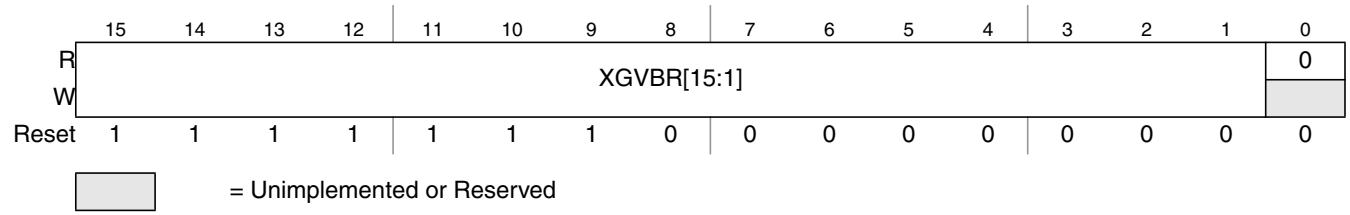



**Figure 10-9. XGATE Vector Base Address Register (XGVBR)**

Read: Anytime

Write: Only if XGATE requests are disabled (XGE = 0) and idle (XGCHID = $00))

**Table 10-9. XGVBR Field Descriptions**

| Field | Description |
|---|---|
| 15–1<br>XBVBR[15:1] | **Vector Base Address** — The XGVBR register holds the start address of the vector block in the XGATE memory map. |

### 10.3.1.8 XGATE Channel Interrupt Flag Vector (XGIF)

The XGATE Channel Interrupt Flag Vector (Figure 10-10) provides access to the interrupt flags of all channels. Each flag may be cleared by writing a "1" to its bit location. Refer to Section 10.5.2, “Outgoing Interrupt Requests” for further information.

Module Base +0x0008

| | 127 | 126 | 125 | 124 | 123 | 122 | 121 | 120 | 119 | 118 | 117 | 116 | 115 | 114 | 113 | 112 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | XGIF_78 | XGF_77 | XGIF_76 | XGIF_75 | XGIF_74 | XGIF_73 | XGIF_72 | XGIF_71 | XGIF_70 |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| | 111 | 110 | 109 | 108 | 107 | 106 | 105 | 104 | 103 | 102 | 101 | 100 | 99 | 98 | 97 | 96 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | XGIF_6F | XGIF_6E | XGIF_6D | XGIF_6C | XGIF_6B | XGIF_6A | XGIF_69 | XGIF_68 | XGF_67 | XGIF_66 | XGIF_65 | XGIF_64 | XGIF_63 | XGIF_62 | XGIF_61 | XGIF_60 |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| | 95 | 94 | 93 | 92 | 91 | 90 | 89 | 88 | 87 | 86 | 85 | 84 | 83 | 82 | 81 | 80 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | XGIF_5F | XGIF_5E | XGIF_5D | XGIF_5C | XGIF_5B | XGIF_5A | XGIF_59 | XGIF_58 | XGF_57 | XGIF_56 | XGIF_55 | XGIF_54 | XGIF_53 | XGIF_52 | XGIF_51 | XGIF_50 |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| | 79 | 78 | 77 | 76 | 75 | 74 | 73 | 72 | 71 | 70 | 69 | 68 | 67 | 66 | 65 | 64 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | XGIF_4F | XGIF_4E | XGIF_4D | XGIF_4C | XGIF_4B | XGIF_4A | XGIF_49 | XGIF_48 | XGF _47 | XGIF_46 | XGIF_45 | XGIF_44 | XGIF_43 | XGIF_42 | XGIF_41 | XGIF_40 |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 10-10. XGATE Channel Interrupt Flag Vector (XGIF)**

| | 63 | 62 | 61 | 60 | 59 | 58 | 57 | 56 | 55 | 54 | 53 | 52 | 51 | 50 | 49 | 48 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | XGIF_3F | XGIF_3E | XGIF_3D | XGIF_3C | XGIF_3B | XGIF_3A | XGIF_39 | XGIF_38 | XGF _37 | XGIF_36 | XGIF_35 | XGIF_34 | XGIF_33 | XGIF_32 | XGIF_31 | XGIF_30 |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| | 47 | 46 | 45 | 44 | 43 | 42 | 41 | 40 | 39 | 38 | 37 | 36 | 35 | 34 | 33 | 32 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | XGIF_2F | XGIF_2E | XGIF_2D | XGIF_2C | XGIF_2B | XGIF_2A | XGIF_29 | XGIF_28 | XGF _27 | XGIF_26 | XGIF_25 | XGIF_24 | XGIF_23 | XGIF_22 | XGIF_21 | XGIF_20 |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | XGIF_1F | XGIF_1E | XGIF_1D | XGIF_1C | XGIF_1B | XGIF_1A | XGIF_19 | XGIF_18 | XGF _17 | XGIF_16 | XGIF_15 | XGIF_14 | XGIF_13 | XGIF_12 | XGIF_11 | XGIF_10 |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | XGIF_0F | XGIF_0E | XGIF_0D | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented or Reserved

**Figure 10-10. XGATE Channel Interrupt Flag Vector (XGIF) (continued)**

Read: Anytime

Write: Anytime

**Table 10-10. XGIV Field Descriptions**

| Field | Description |
|---|---|
| 127–9<br>XGIF[78:9] | **Channel Interrupt Flags** — These bits signal pending channel interrupts. They can only be set by the RISC core (see SIF instruction on page 10-449). Each flag can be cleared by writing a "1" to its bit location. Unimplemented interrupt flags will always read "0". Section “Interrupts” of the **device overview** for a list of implemented Interrupts.<br>Read:<br>0 Channel interrupt is not pending<br>1 Channel interrupt is pending if XGIE is set<br>Write:<br>0 No effect<br>1 Clears the interrupt flag |

**NOTE**

Suggested Mnemonics for accessing the interrupt flag vector on a word basis are:

**XGIF_7F_70** (XGIF[127:112]),
**XGIF_6F_60** (XGIF[111:96]),
**XGIF_5F_50** (XGIF[95:80]),
**XGIF_4F_40** (XGIF[79:64]),
**XGIF_3F_30** (XGIF[63:48]),
**XGIF_2F_20** (XGIF[47:32]),
**XGIF_1F_10** (XGIF[31:16]),
**XGIF_0F_00** (XGIF[15:0])

### 10.3.1.9 XGATE Software Trigger Register (XGSWT)

The eight software triggers of the XGATE module can be set and cleared through the XGATE Software Trigger Register (Figure 10-11). The upper byte of this register, the software trigger mask, controls the write access to the lower byte, the software trigger bits. These bits can be set or cleared if a "1" is written to the associated mask in the same bus cycle. Refer to Section 10.5.2, “Outgoing Interrupt Requests” for further information.

Module Base +0x00018

| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | XGSWT[7:0] | | | | | | | |
| W | XGSWTM[7:0] | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 10-11. XGATE Software Trigger Register (XGSWT)**

Read: Anytime

Write: Anytime

**Table 10-11. XGSWT Field Descriptions**

| Field | Description |
|---|---|
| 15–8<br>XGSWTM[7:0] | **Software Trigger Mask** — These bits control the write access to the XGSWT bits. Each XGSWT bit can only be written if a "1" is written to the corresponding XGSWTM bit in the same access.<br>Read:<br>These bits will always read "0".<br>Write:<br>0 Disable write access to the XGSWT in the same bus cycle<br>1 Enable write access to the corresponding XGSWT bit in the same bus cycle |
| 7–0<br>XGSWT[7:0] | **Software Trigger Bits** — These bits act as interrupt flags that are able to trigger XGATE software channels. They can only be set and cleared by software.<br>Read:<br>0 No software trigger pending<br>1 Software trigger pending if the XGIE bit is set<br>Write:<br>0 Clear Software Trigger<br>1 Set Software Trigger |

**NOTE**

The XGATE channel IDs that are associated with the eight software triggers are determined on chip integration level. (see Section “Interrupts“ of the **device overview**)

XGATE software triggers work like any peripheral interrupt. They can be used as XGATE requests as well as S12X_CPU interrupts. The target of the software trigger must be selected in the S12X_INT module.

### 10.3.1.10 XGATE Semaphore Register (XGSEM)

The XGATE provides a set of eight hardware semaphores that can be shared between the S12X_CPU and the XGATE RISC core. Each semaphore can either be unlocked, locked by the S12X_CPU or locked by the RISC core. The RISC core is able to lock and unlock a semaphore through its SSEM and CSEM instructions. The S12X_CPU has access to the semaphores through the XGATE Semaphore Register (Figure 10-12). Refer to section Section 10.4.4, “Semaphores” for details.

Module Base +0x0001A

| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | XGSEM[7:0] | | | | | | | |
| W | XGSEMM[7:0] | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 10-12. XGATE Semaphore Register (XGSEM)**

Read: Anytime

Write: Anytime (see Section 10.4.4, “Semaphores”)

**Table 10-12. XGSEM Field Descriptions**

| Field | Description |
|---|---|
| 15–8<br>XGSEMM[7:0] | **Semaphore Mask** — These bits control the write access to the XGSEM bits.<br>Read:<br>These bits will always read "0".<br>Write:<br>0 Disable write access to the XGSEM in the same bus cycle<br>1 Enable write access to the XGSEM in the same bus cycle |
| 7–0<br>XGSEM[7:0] | **Semaphore Bits** — These bits indicate whether a semaphore is locked by the S12X_CPU. A semaphore can be attempted to be set by writing a "1" to the XGSEM bit and to the corresponding XGSEMM bit in the same write access. Only unlocked semaphores can be set. A semaphore can be cleared by writing a "0" to the XGSEM bit and a "1" to the corresponding XGSEMM bit in the same write access.<br>Read:<br>0 Semaphore is unlocked or locked by the RISC core<br>1 Semaphore is locked by the S12X_CPU<br>Write:<br>0 Clear semaphore if it was locked by the S12X_CPU<br>1 Attempt to lock semaphore by the S12X_CPU |

### 10.3.1.11 XGATE Condition Code Register (XGCCR)

The XGCCR register (Figure 10-13) provides access to the RISC core's condition code register.

Module Base +0x001D

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | 0 | 0 | XGN | XGZ | XGV | XGC |
| W | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented or Reserved

**Figure 10-13. XGATE Condition Code Register (XGCCR)**

Read: In debug mode if unsecured and not idle (XGCHID ≠ 0x00)

Write: In debug mode if unsecured and not idle (XGCHID ≠ 0x00)

**Table 10-13. XGCCR Field Descriptions**

| Field | Description |
|---|---|
| 3<br>XGN | **Sign Flag** — The RISC core's Sign flag |
| 2<br>XGZ | **Zero Flag** — The RISC core's Zero flag |
| 1<br>XGV | **Overflow Flag** — The RISC core's Overflow flag |
| 0<br>XGC | **Carry Flag** — The RISC core's Carry flag |

### 10.3.1.12 XGATE Program Counter Register (XGPC)

The XGPC register (Figure 10-14) provides access to the RISC core's program counter.

Module Base +0x0001E

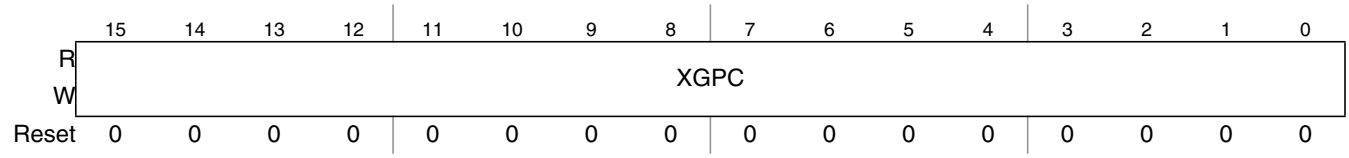



**Figure 10-14. XGATE Program Counter Register (XGPC)**

Read: In debug mode if unsecured and not idle (XGCHID ≠ 0x00)

Write: In debug mode if unsecured and not idle (XGCHID ≠ 0x00)

**Table 10-14. XGPC Field Descriptions**

| Field | Description |
|---|---|
| 15–0<br>XGPC[15:0] | **Program Counter** — The RISC core's program counter |

### 10.3.1.13 XGATE Register 1 (XGR1)

The XGR1 register (Figure 10-15) provides access to the RISC core's register 1.

Module Base +0x00022

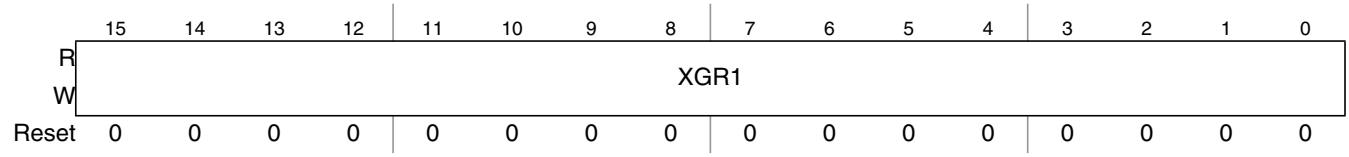



**Figure 10-15. XGATE Register 1 (XGR1)**

Read: In debug mode if unsecured and not idle (XGCHID ≠ 0x00)

Write: In debug mode if unsecured and not idle (XGCHID ≠ 0x00)

**Table 10-15. XGR1 Field Descriptions**

| Field | Description |
|---|---|
| 15–0<br>XGR1[15:0] | **XGATE Register 1** — The RISC core's register 1 |

### 10.3.1.14 XGATE Register 2 (XGR2)

The XGR2 register (Figure 10-16) provides access to the RISC core's register 2.

Module Base +0x00024

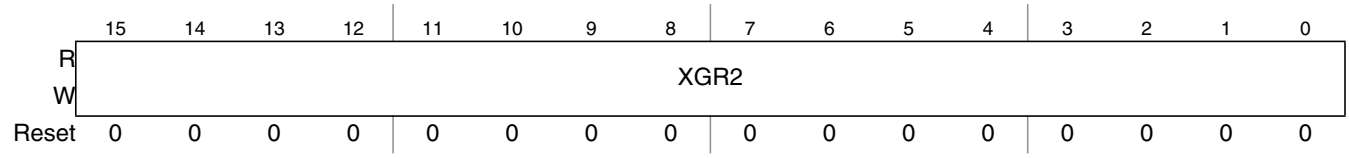



**Figure 10-16. XGATE Register 2 (XGR2)**

Read: In debug mode if unsecured and not idle (XGCHID ≠ 0x00)

Write: In debug mode if unsecured and not idle (XGCHID ≠ 0x00)

**Table 10-16. XGR2 Field Descriptions**

| Field | Description |
|---|---|
| 15–0<br>XGR2[15:0] | **XGATE Register 2** — The RISC core's register 2 |

### 10.3.1.15 XGATE Register 3 (XGR3)

The XGR3 register (Figure 10-17) provides access to the RISC core's register 3.

Module Base +0x00026

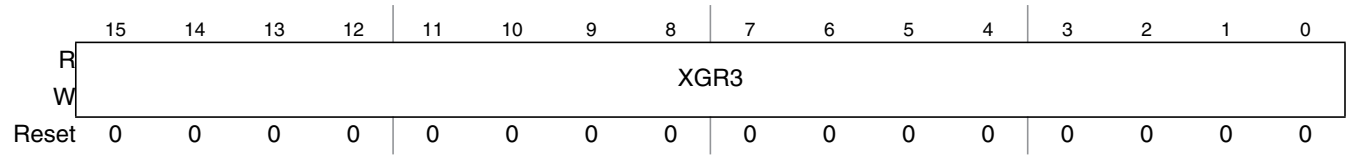



**Figure 10-17. XGATE Register 3 (XGR3)**

Read: In debug mode if unsecured and not idle (XGCHID ≠ 0x00)

Write: In debug mode if unsecured and not idle (XGCHID ≠ 0x00)

**Table 10-17. XGR3 Field Descriptions**

| Field | Description |
|---|---|
| 15–0<br>XGR3[15:0] | **XGATE Register 3** — The RISC core's register 3 |

### 10.3.1.16 XGATE Register 4 (XGR4)

The XGR4 register (Figure 10-18) provides access to the RISC core's register 4.

Module Base +0x00028

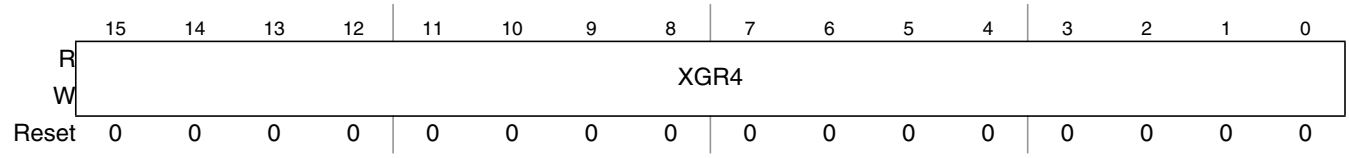



**Figure 10-18. XGATE Register 4 (XGR4)**

Read: In debug mode if unsecured and not idle (XGCHID ≠ 0x00)

Write: In debug mode if unsecured and not idle (XGCHID ≠ 0x00)

**Table 10-18. XGR4 Field Descriptions**

| Field | Description |
|---|---|
| 15–0<br>XGR4[15:0] | **XGATE Register 4** — The RISC core's register 4 |

### 10.3.1.17 XGATE Register 5 (XGR5)

The XGR5 register (Figure 10-19) provides access to the RISC core's register 5.

Module Base +0x0002A

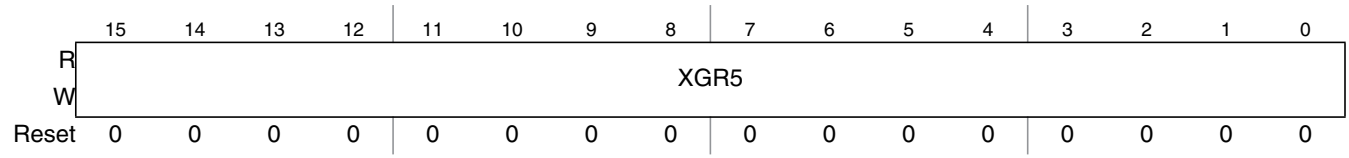



**Figure 10-19. XGATE Register 5 (XGR5)**

Read: In debug mode if unsecured and not idle (XGCHID ≠ 0x00)

Write: In debug mode if unsecured and not idle (XGCHID ≠ 0x00)

**Table 10-19. XGR5 Field Descriptions**

| Field | Description |
|---|---|
| 15–0<br>XGR5[15:0] | **XGATE Register 5** — The RISC core's register 5 |

### 10.3.1.18 XGATE Register 6 (XGR6)

The XGR6 register (Figure 10-20) provides access to the RISC core's register 6.

Module Base +0x0002C

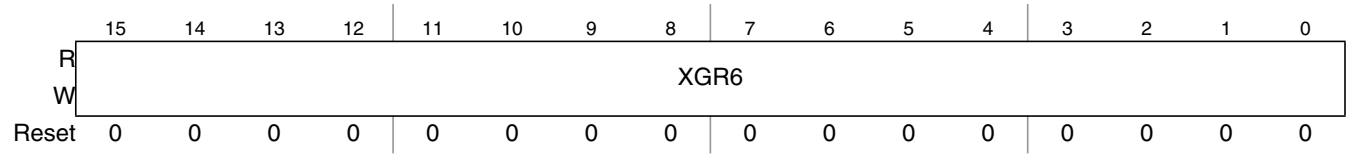



**Figure 10-20. XGATE Register 6 (XGR6)**

Read: In debug mode if unsecured and not idle (XGCHID ≠ 0x00)

Write: In debug mode if unsecured and not idle (XGCHID ≠ 0x00)

**Table 10-20. XGR6 Field Descriptions**

| Field | Description |
|---|---|
| 15–0<br>XGR6[15:0] | **XGATE Register 6** — The RISC core's register 6 |

### 10.3.1.19 XGATE Register 7 (XGR7)

The XGR7 register (Figure 10-21) provides access to the RISC core's register 7.

Module Base +0x0002E

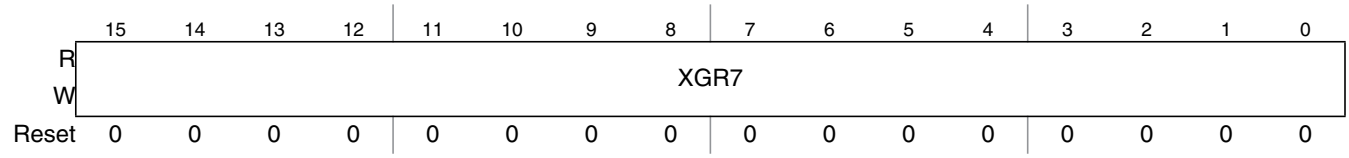



**Figure 10-21. XGATE Register 7 (XGR7)**

Read: In debug mode if unsecured and not idle (XGCHID ≠ 0x00)

Write: In debug mode if unsecured and not idle (XGCHID ≠ 0x00)

**Table 10-21. XGR7 Field Descriptions**

| Field | Description |
|---|---|
| 15–0<br>XGR7[15:0] | **XGATE Register 7** — The RISC core's register 7 |

## 10.4 Functional Description

The core of the XGATE module is a RISC processor which is able to access the MCU's internal memories and peripherals (see Figure 10-1). The RISC processor always remains in an idle state until it is triggered by an XGATE request. Then it executes a code sequence (thread) that is associated with the requested XGATE channel. Each thread can run on a priority level ranging from 1 to 7. Refer to the **S12X_INT**

**Section** for information on how to select priority levels for XGATE threads. Low priority threads (interrupt levels 1 to 3) can be interrupted by high priority threads (interrupt levels 4 to 7). High priority threads are not interruptible. The register content of an interrupted thread is maintained and restored by the XGATE hardware.

To signal the completion of a task the XGATE is able to send interrupts to the S12X_CPU. Each XGATE channel has its own interrupt vector. Refer to the **S12X_INT Section** for detailed information.

The XGATE module also provides a set of hardware semaphores which are necessary to ensure data consistency whenever RAM locations or peripherals are shared with the S12X_CPU.

The following sections describe the components of the XGATE module in further detail.

## 10.4.1 XGATE RISC Core

The RISC core is a 16 bit processor with an instruction set that is well suited for data transfers, bit manipulations, and simple arithmetic operations (see Section 10.8, “Instruction Set”).

It is able to access the MCU’s internal memories and peripherals without blocking these resources from the S12X_CPU[1]. Whenever the S12X_CPU and the RISC core access the same resource, the RISC core will be stalled until the resource becomes available again.[1]

The XGATE offers a high access rate to the MCU’s internal RAM. Depending on the bus load, the RISC core can perform up to two RAM accesses per S12X_CPU bus cycle.

Bus accesses to peripheral registers or flash are slower. A transfer rate of one bus access per S12X_CPU cycle can not be exceeded.

The XGATE module is intended to execute short interrupt service routines that are triggered by peripheral modules or by software.

## 10.4.2 Programmer’s Model

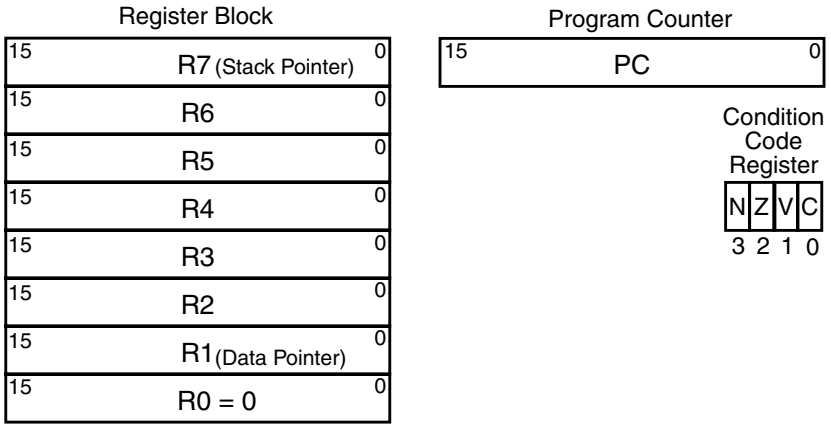



**Figure 10-22. Programmer’s Model**

1. With the exception of PRR registers (see Section “S12X_MMC”).

The programmer's model of the XGATE RISC core is shown in Figure 10-22. The processor offers a set of seven general purpose registers (R1 - R7), which serve as accumulators and index registers. An additional eighth register (R0) is tied to the value "$0000". Registers R1 and R7 have additional functionality. R1 is preloaded with the initial data pointer of the channel's service request vector (see Figure 10-23). R7 is either preloaded with the content of XGISP74 if the interrupt priority of the current channel is in the range 7 to 4, or it is with preloaded the content of XGISP31 if the interrupt priority of the current channel is in the range 3 to 1. The remaining general purpose registers will be reset to an unspecified value at the beginning of each thread.

The 16 bit program counter allows the addressing of a 64 kbyte address space.

The condition code register contains four bits: the sign bit (S), the zero flag (Z), the overflow flag (V), and the carry bit (C). The initial content of the condition code register is undefined.

### 10.4.3 Memory Map

The XGATE's RISC core is able to access an address space of 64K bytes. The allocation of memory blocks within this address space is determined on chip level. Refer to the **S12X_MMC Section** for a detailed information.

The XGATE vector block assigns a start address and a data pointer to each XGATE channel. Its position in the XGATE memory map can be adjusted through the XGVBR register (see Section 10.3.1.7, "XGATE Vector Base Address Register (XGVBR)"). Figure 10-23 shows the layout of the vector block. Each vector consists of two 16 bit words. The first contains the start address of the service routine. This value will be loaded into the program counter before a service routine is executed. The second word is a pointer to the service routine's data space. This value will be loaded into register R1 before a service routine is executed.

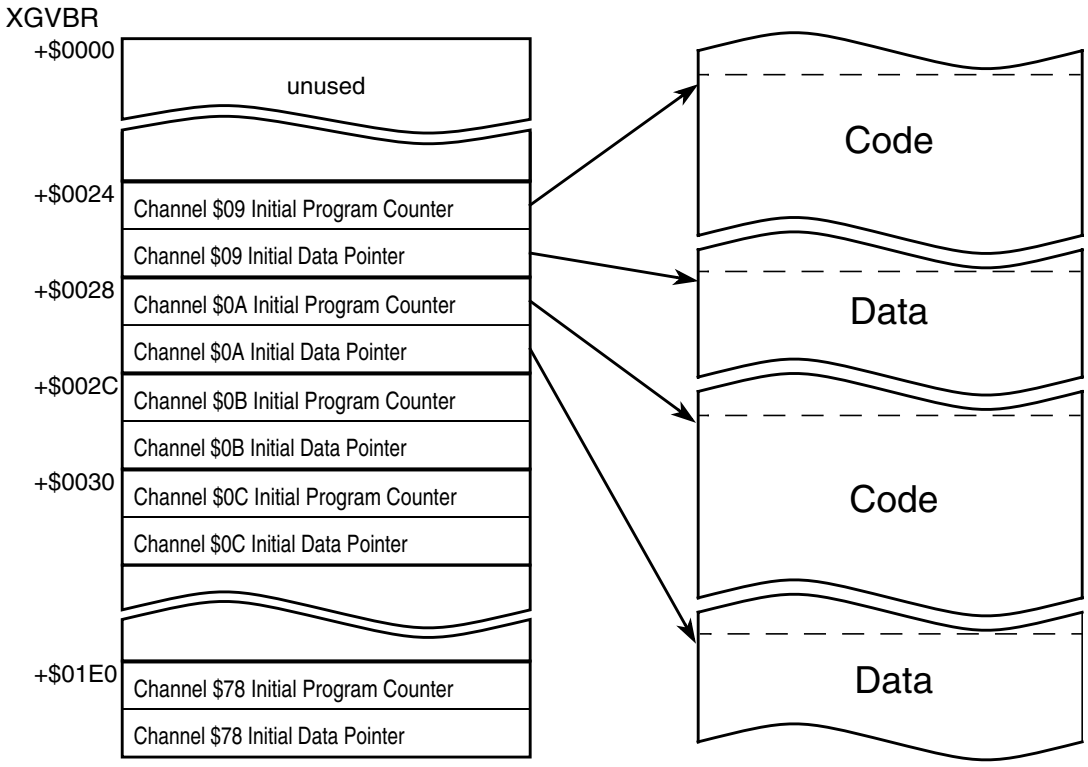



**Figure 10-23. XGATE Vector Block**

## 10.4.4 Semaphores

The XGATE module offers a set of eight hardware semaphores. These semaphores provide a mechanism to protect system resources that are shared between two concurrent threads of program execution; one thread running on the S12X_CPU and one running on the XGATE RISC core.

Each semaphore can only be in one of the three states: “Unlocked”, “Locked by S12X_CPU”, and “Locked by XGATE”. The S12X_CPU can check and change a semaphore’s state through the XGATE semaphore register (XGSEM, see Section 10.3.1.10, “XGATE Semaphore Register (XGSEM)”). The RISC core does this through its SSEM and CSEM instructions.

IFigure 10-24 illustrates the valid state transitions.

**set_xgsem**: 1 is written to XGSEM[*n*] (and 1 is written to XGSEMM[*n*])
**clr_xgsem**: 0 is written to XGSEM[*n*] (and 1 is written to XGSEMM[*n*])
**ssem**: Executing SSEM instruction (on semaphore *n*)
**csem**: Executing CSEM instruction (on semaphore *n*)

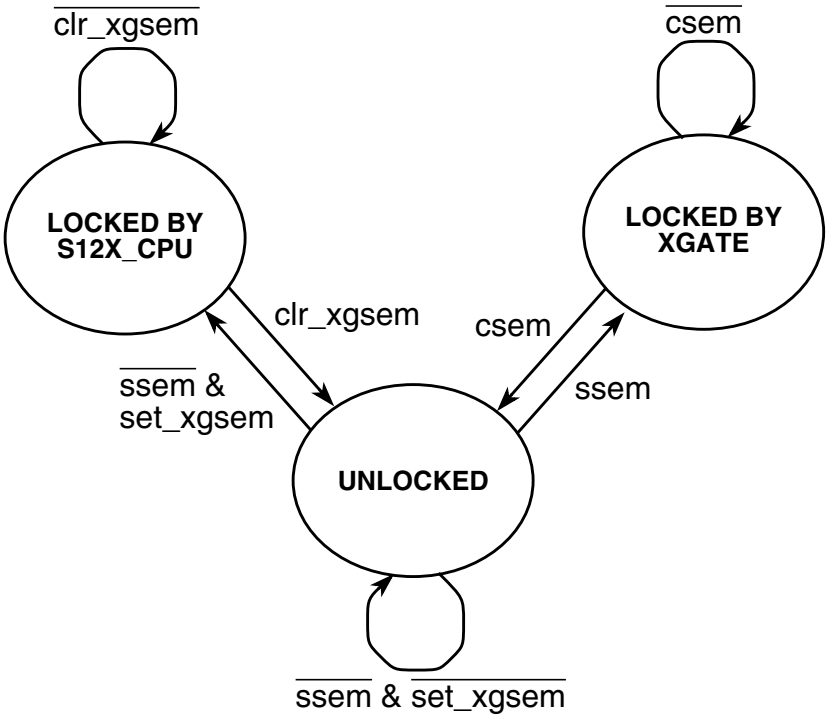



**Figure 10-24. Semaphore State Transitions**

Figure 10-25 gives an example of the typical usage of the XGATE hardware semaphores.

Two concurrent threads are running on the system. One is running on the S12X_CPU and the other is running on the RISC core. They both have a critical section of code that accesses the same system resource. To guarantee that the system resource is only accessed by one thread at a time, the critical code sequence must be embedded in a semaphore lock/release sequence as shown.

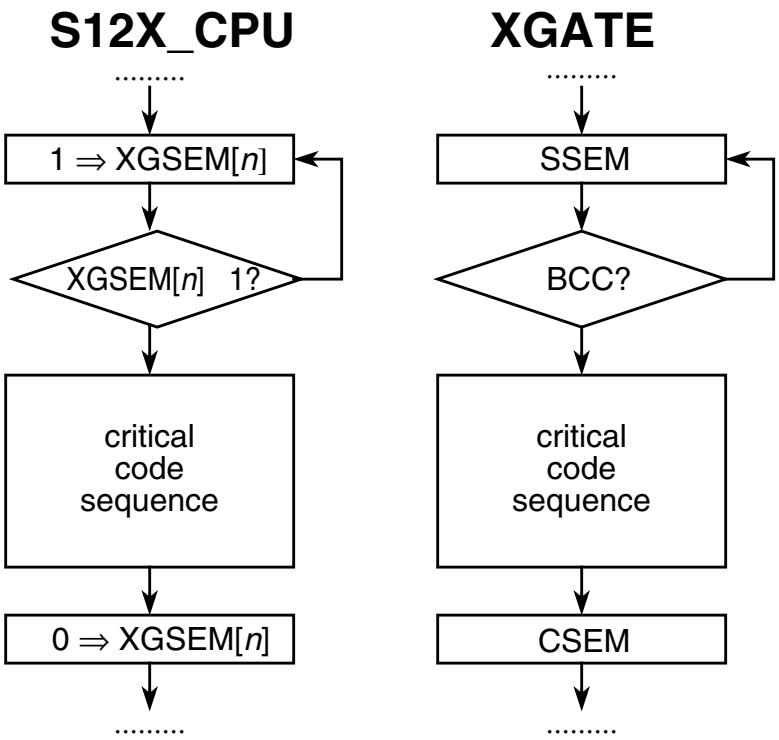



**Figure 10-25. Algorithm for Locking and Releasing Semaphores**

## 10.4.5 Software Error Detection

Upon detecting an error condition caused by erratic application code, the XGATE module will immediately terminate program execution and trigger a non-maskable interrupt to the S12X_CPU. There are three error conditions:

- Execution of an illegal opcode
- Illegal opcode fetches
- Illegal load or store accesses

All opcodes which are not listed in section Section 10.8, “Instruction Set” are illegal opcodes. Illegal opcode fetches as well as illegal load and store accesses are defined on chip level. Refer to the **S12X_MMC Section** for a detailed information.

**NOTE**

When executing a branch (BCC, BCS,...), a jump (JAL) or an RTS instruction, the XGATE prefetches and discards the opcode of the following instruction. The XGATE will perform its software error handling actions (see above) if this opcode fetch is illegal.

## 10.5 Interrupts

### 10.5.1 Incoming Interrupt Requests

XGATE threads are triggered by interrupt requests which are routed to the XGATE module (see S12X_INT Section). Only a subset of the MCU's interrupt requests can be routed to the XGATE. Which specific interrupt requests these are and which channel ID they are assigned to is documented in Section "Interrupts" of the **device overview**.

### 10.5.2 Outgoing Interrupt Requests

There are three types of interrupt requests which can be triggered by the XGATE module:

4. Channel interrupts

   For each XGATE channel there is an associated interrupt flag in the XGATE interrupt flag vector (XGIF, see Section 10.3.1.8, "XGATE Channel Interrupt Flag Vector (XGIF)"). These flags can be set through the "SIF" instruction by the RISC core. They are typically used to flag an interrupt to the S12X_CPU when the XGATE has completed one of its task.

5. Software triggers

   Software triggers are interrupt flags, which can be set and cleared by software (see Section 10.3.1.9, "XGATE Software Trigger Register (XGSWT)"). They are typically used to trigger XGATE tasks by the S12X_CPU software. However these interrupts can also be routed to the S12X_CPU (see **S12X_INT Section**) and triggered by the XGATE software.

6. Software error interrupt

   The software error interrupt signals to the S12X_CPU the detection of an error condition in the XGATE application code (see Section 10.4.5, "Software Error Detection"). This is a non-maskable interrupt. Executing the interrupt service routine will automatically reset the interrupt line.

All outgoing XGATE interrupts, except software error interrupts, can be disabled by the XGIE bit in the XGATE module control register (XGMCTL, see Section 10.3.1.1, "XGATE Control Register (XGMCTL)").

## 10.6 Debug Mode

The XGATE debug mode is a feature to allow debugging of application code.

### 10.6.1 Debug Features

In debug mode the RISC core will be halted and the following debug features will be enabled:

- Read and Write accesses to RISC core registers (XGCCR, XGPC, XGR1–XGR7)[1]

  All RISC core registers can be modified. Leaving debug mode will cause the RISC core to continue program execution with the modified register values.

1. Only possible if MCU is unsecured

- Single Stepping

  Writing a "1" to the XGSS bit will call the RISC core to execute a single instruction. All RISC core registers will be updated accordingly.

- Write accesses to the XGCHID register and the XGCHPL register

  XGATE threads can be initiated and terminated through a 16 write access to the XGCHID and the XGCHPL register or through a 8 bit write access to the XGCHID register. Detailed operation is shown in Table 10-22. Once a thread has been initiated it's code can be either single stepped or it can be executed by leaving debug mode.

**Table 10-22. Initiating and Terminating Threads in Debug Mode**

| Register Content | | Single Cycle Write Access to... | | Action |
|---|---|---|---|---|
| XGCHID | XGCHPL | XGCHID | XGCHPL | |
| 0 | 0 | 1..127 | -(1) | Set new XGCHID<br>Set XGCHPL to 0x01<br>Initiate new thread |
| 0 | 0 | 1..127 | 0..7 | Set new XGCHID<br>Set new XGCHPL<br>Initiate new thread |
| 1..127 | 0..3 | 1..127 | 4..7 | Interrupt current thread<br>Set new XGCHID<br>Set new XGCHPL<br>Initiate new thread |
| 1..127 | 0..7 | 0 | 0..7 | Terminate current thread.<br>Resume interrupted thread or become idle if no interrupted thread is pending |
| | | | -1 | |
| All other combinations | | | | No action |

1. 8 bit write access to XGCHID

**NOTE**

Even though zero is not a valid interrupt priority level of the S12X_INT module, a thread of priority level 0 can be initiated in debug mode. The XGATE handles requests of priority level 0 in the same way as it handles requests of priority levels 1 to 3.

**NOTE**

All channels 1 to 127 can be initiated by writing to the XGCHID register, even if they are not assigned to any peripheral module.

**NOTE**

In Debug Mode the XGATE will ignore all requests from peripheral modules.

#### 10.6.1.0.1 Entering Debug Mode

Debug mode can be entered in four ways:

1. Setting XGDBG to "1"

Writing a "1" to XGDBG and XGDBGM in the same write access causes the XGATE to enter debug mode upon completion of the current instruction.

**NOTE**

After writing to the XGDBG bit the XGATE will not immediately enter debug mode. Depending on the instruction that is executed at this time there may be a delay of several clock cycles. The XGDBG will read "0" until debug mode is entered.

2. Software breakpoints

   XGATE programs which are stored in the internal RAM allow the use of software breakpoints. A software breakpoint is set by replacing an instruction of the program code with the "BRK" instruction.

   As soon as the program execution reaches the "BRK" instruction, the XGATE enters debug mode. Additionally a software breakpoint request is sent to the S12X_DBG module (see section 4.9 of the **S12X_DBG Section**).

   Upon entering debug mode, the program counter will point to the "BRK" instruction. The other RISC core registers will hold the result of the previous instruction.

   To resume program execution, the "BRK" instruction must be replaced by the original instruction before leaving debug mode.

3. Tagged Breakpoints

   The S12X_DBG module is able to place tags on fetched opcodes. The XGATE is able to enter debug mode right before a tagged opcode is executed (see section 4.9 of the **S12X_DBG Section**). Upon entering debug mode, the program counter will point to the tagged instruction. The other RISC core registers will hold the result of the previous instruction.

4. Forced Breakpoints

   Forced breakpoints are triggered by the S12X_DBG module (see section 4.9 of the **S12X_DBG Section**). When a forced breakpoint occurs, the XGATE will enter debug mode upon completion of the current instruction.

### 10.6.2 Leaving Debug Mode

Debug mode can only be left by setting the XGDBG bit to "0". If a thread is active (XGCHID has not been cleared in debug mode), program execution will resume at the value of XGPC.

## 10.7 Security

In order to protect XGATE application code on secured S12X devices, a few restrictions in the debug features have been made. These are:

- Registers XGCCR, XGPC, and XGR1–XGR7 will read zero on a secured device
- Registers XGCCR, XGPC, and XGR1–XGR7 can not be written on a secured device
- Single stepping is not possible on a secured device

# 10.8 Instruction Set

## 10.8.1 Addressing Modes

For the ease of implementation the architecture is a strict Load/Store RISC machine, which means all operations must have one of the eight general purpose registers R0 … R7 as their source as well their destination.

All word accesses must work with a word aligned address, that is A[0] = 0!

### 10.8.1.1 Naming Conventions

| | |
|---|---|
| RD | Destination register, allowed range is R0–R7 |
| RD.L | Low byte of the destination register, bits [7:0] |
| RD.H | High byte of the destination register, bits [15:8] |
| RS, RS1, RS2 | Source register, allowed range is R0–R7 |
| RS.L, RS1.L, RS2.L | Low byte of the source register, bits [7:0] |
| RS.H, RS1.H, RS2.H | High byte of the source register, bits[15:8] |
| RB | Base register for indexed addressing modes, allowed range is R0–R7 |
| RI | Offset register for indexed addressing modes with register offset, allowed range is R0–R7 |
| RI+ | Offset register for indexed addressing modes with register offset and post-increment, Allowed range is R0–R7 (R0+ is equivalent to R0) |
| –RI | Offset register for indexed addressing modes with register offset and pre-decrement, Allowed range is R0–R7 (–R0 is equivalent to R0) |

**NOTE**

Even though register R1 is intended to be used as a pointer to the data segment, it may be used as a general purpose data register as well.

Selecting R0 as destination register will discard the result of the instruction. Only the condition code register will be updated

### 10.8.1.2 Inherent Addressing Mode (INH)

Instructions that use this addressing mode either have no operands or all operands are in internal XGATE registers.

Examples:

```
BRK
RTS
```

### 10.8.1.3 Immediate 3-Bit Wide (IMM3)

Operands for immediate mode instructions are included in the instruction stream and are fetched into the instruction queue along with the rest of the 16 bit instruction. The '#' symbol is used to indicate an immediate addressing mode operand. This address mode is used for semaphore instructions.

Examples:

```
CSEM    #1      ; Unlock semaphore 1
SSEM    #3      ; Lock Semaphore 3
```

### 10.8.1.4 Immediate 4 Bit Wide (IMM4)

The 4 bit wide immediate addressing mode is supported by all shift instructions.

RD = RD ∗ IMM4

Examples:

```
LSL     R4,#1   ; R4 = R4 << 1; shift register R4 by 1 bit to the left
LSR     R4,#3   ; R4 = R4 >> 3; shift register R4 by 3 bits to the right
```

### 10.8.1.5 Immediate 8 Bit Wide (IMM8)

The 8 bit wide immediate addressing mode is supported by four major commands (ADD, SUB, LD, CMP).

RD = RD ∗ imm8

Examples:

```
ADDL    R1,#1   ; adds an 8 bit value to register R1
SUBL    R2,#2   ; subtracts an 8 bit value from register R2
LDH     R3,#3   ; loads an 8 bit immediate into the high byte of Register R3
CMPL    R4,#4   ; compares the low byte of register R4 with an immediate value
```

### 10.8.1.6 Immediate 16 Bit Wide (IMM16)

The 16 bit wide immediate addressing mode is a construct to simplify assembler code. Instructions which offer this mode are translated into two opcodes using the eight bit wide immediate addressing mode.

RD = RD ∗ IMM16

Examples:

```
LDW     R4,#$1234       ; translated to LDL R4,#$34; LDH R4,#$12
ADD     R4,#$5678       ; translated to ADDL R4,#$78; ADDH R4,#$56
```

### 10.8.1.7 Monadic Addressing (MON)

In this addressing mode only one operand is explicitly given. This operand can either be the source (*f*(RD)), the target (RD = *f*()), or both source and target of the operation (RD = *f*(RD)).

Examples:

```
JAL     R1      ; PC = R1, R1 = PC+2
SIF     R2      ; Trigger IRQ associated with the channel number in R2.L
```

### 10.8.1.8 Dyadic Addressing (DYA)

In this mode the result of an operation between two registers is stored in one of the registers used as operands.

RD = RD ∗ RS is the general register to register format, with register RD being the first operand and RS the second. RD and RS can be any of the 8 general purpose registers R0 … R7. If R0 is used as the destination register, only the condition code flags are updated. This addressing mode is used only for shift operations with a variable shift value

Examples:

```
LSL     R4,R5    ; R4 = R4 << R5
LSR     R4,R5    ; R4 = R4 >> R5
```

### 10.8.1.9 Triadic Addressing (TRI)

In this mode the result of an operation between two or three registers is stored into a third one.
RD = RS1 ∗ RS2 is the general format used in the order RD, RS1, RS1. RD, RS1, RS2 can be any of the 8 general purpose registers R0 … R7. If R0 is used as the destination register RD, only the condition code flags are updated. This addressing mode is used for all arithmetic and logical operations.

Examples:

```
ADC     R5,R6,R7          ; R5 = R6 + R7 + Carry
SUB     R5,R6,R7          ; R5 = R6 - R7
```

### 10.8.1.10 Relative Addressing 9-Bit Wide (REL9)

A 9-bit signed word address offset is included in the instruction word. This addressing mode is used for conditional branch instructions.

Examples:

```
BCC     REL9              ; PC = PC + 2 + (REL9 << 1)
BEQ     REL9              ; PC = PC + 2 + (REL9 << 1)
```

### 10.8.1.11 Relative Addressing 10-Bit Wide (REL10)

An 10-bit signed word address offset is included in the instruction word. This addressing mode is used for the unconditional branch instruction.

Examples:

```
BRA     REL10             ; PC = PC + 2 + (REL10 << 1)
```

### 10.8.1.12 Index Register plus Immediate Offset (IDO5)

(RS, #OFFS5) provides an unsigned offset from the base register.

Examples:

```
LDB     R4,(R1,#OFFS5)    ; loads a byte from (R1+OFFS5) into R4
STW     R4,(R1,#OFFS5)    ; stores R4 as a word to (R1+OFFS5)
```

#### 10.8.1.13 Index Register plus Register Offset (IDR)

For load and store instructions (RS, RI) provides a variable offset in a register.

Examples:

```
LDB     R4,(R1,R2)        ; loads a byte from (R1+R2) into R4
STW     R4,(R1,R2)        ; stores R4 as a word to (R1+R2)
```

#### 10.8.1.14 Index Register plus Register Offset with Post-increment (IDR+)

[RS, RI+] provides a variable offset in a register, which is incremented after accessing the memory. In case of a byte access the index register will be incremented by one. In case of a word access it will be incremented by two.

Examples:

```
LDB     R4,(R1,R2+)       ; loads a byte from (R1+R2) into R4, R2+=1
STW     R4,(R1,R2+)       ; stores R4 as a word to (R1+R2), R2+=2
```

#### 10.8.1.15 Index Register plus Register Offset with Pre-decrement (–IDR)

[RS, -RI] provides a variable offset in a register, which is decremented before accessing the memory. In case of a byte access the index register will be decremented by one. In case of a word access it will be decremented by two.

Examples:

```
LDB     R4,(R1,-R2)       ; R2 -=1, loads a byte from (R1+R2) into R4
STW     R4,(R1,-R2)       ; R2 -=2, stores R4 as a word to (R1+R2)
```

### 10.8.2 Instruction Summary and Usage

#### 10.8.2.1 Load & Store Instructions

Any register can be loaded either with an immediate or from the address space using indexed addressing modes.

```
LDL     RD,#IMM8          ; loads an immediate 8 bit value to the lower byte of RD
LDW     RD,(RB,RI)        ; loads data using RB+RI as effective address

LDB     RD,(RB, RI+)      ; loads data using RB+RI as effective address
                          ; followed by an increment of RI depending on
                          ; the size of the operation
```

The same set of modes is available for the store instructions

```
STB     RS,(RB, RI)       ; stores data using RB+RI as effective address

STW     RS,(RB, RI+)      ; stores data using RB+RI as effective address
                          ; followed by an increment of RI depending on
                          ; the size of the operation.
```

### 10.8.2.2 Logic and Arithmetic Instructions

All logic and arithmetic instructions support the 8 bit immediate addressing mode (IMM8: RD = RD ∗ #IMM8) and the triadic addressing mode (TRI: RD = RS1 ∗ RS2).

All arithmetic is considered as signed, sign, overflow, zero and carry flag will be updated. The carry will not be affected for logical operations.

```
        ADDL    R2,#1           ; increment R2
        ANDH    R4,#$FE         ; R4.H = R4.H & $FE, clear lower bit of higher byte

        ADD     R3,R4,R5        ; R3 = R4 + R5
        SUB     R3,R4,R5        ; R3 = R4 - R5

        AND     R3,R4,R5        ; R3 = R4 & R5 logical AND on the whole word
        OR      R3,R4,R5        ; R3 = R4 | R5
```

### 10.8.2.3 Register – Register Transfers

This group comprises transfers from and to some special registers

```
        TFR     R3,CCR          ; transfers the condition code register to the low byte of
                                ; register R3
Branch Instructions
```

The branch offset is +255 words or -256 words counted from the beginning of the next instruction. Since instructions have a fixed 16 bit width, the branch offsets are word aligned by shifting the offset value by 2.

```
        BEQ     label           ; if Z flag = 1 branch to label
```

An unconditional branch allows a +511 words or -512 words branch distance.

```
        BRA     label
```

### 10.8.2.4 Shift Instructions

Shift operations allow the use of a 4 bit wide immediate value to identify a shift width within a 16 bit word. For shift operations a value of 0 does not shift at all, while a value of 15 shifts the register RD by 15 bits. In a second form the shift value is contained in the bits 3:0 of the register RS.

Examples:

```
        LSL     R4,#1   ; R4 = R4 << 1; shift register R4 by 1 bit to the left
        LSR     R4,#3   ; R4 = R4 >> 3; shift register R4 by 3 bits to the right
        ASR     R4,R2   ; R4 = R4 >> R2;arithmetic shift register R4 right by the amount
                        ;            of bits contained in R2[3:0].
```

### 10.8.2.5 Bit Field Operations

This addressing mode is used to identify the position and size of a bit field for insertion or extraction. The width and offset are coded in the lower byte of the source register 2, RS2. The content of the upper byte is ignored. An offset of 0 denotes the right most position and a width of 0 denotes 1 bit. These instructions are very useful to extract, insert, clear, set or toggle portions of a 16 bit word

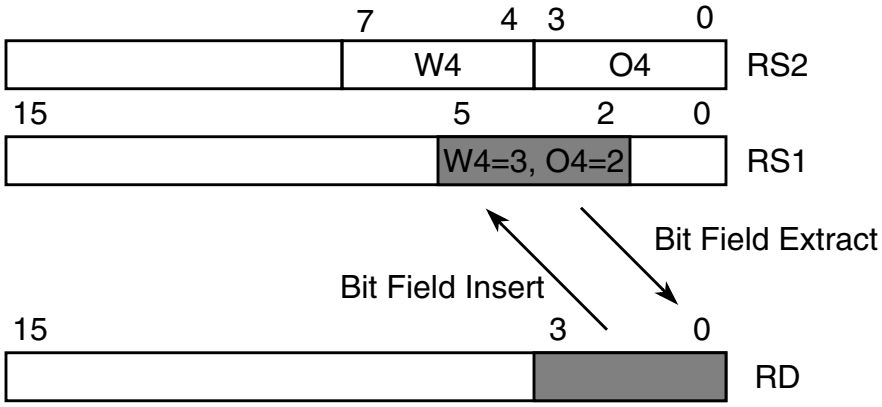



**Figure 10-26. Bit Field Addressing**

```
BFEXT    R3,R4,R5 ; R5: W4+1 bits with offset O4, will be extracted from R4 into R3
```

### 10.8.2.6 Special Instructions for DMA Usage

The XGATE offers a number of additional instructions for flag manipulation, program flow control and debugging:

1. SIF: Set a channel interrupt flag
2. SSEM: Test and set a hardware semaphore
3. CSEM: Clear a hardware semaphore
4. BRK: Software breakpoint
5. NOP: No Operation
6. RTS: Terminate the current thread

## 10.8.3 Cycle Notation

Table 10-23 show the XGATE access detail notation. Each code letter equals one XGATE cycle. Each letter implies additional wait cycles if memories or peripherals are not accessible. Memories or peripherals are not accessible if they are blocked by the S12X_CPU. In addition to this Peripherals are only accessible every other XGATE cycle. Uppercase letters denote 16 bit operations. Lowercase letters denote 8 bit operations. The XGATE is able to perform two bus or wait cycles per S12X_CPU cycle.

**Table 10-23. Access Detail Notation**

| |
|---|
| V — Vector fetch: always an aligned word read, lasts for at least one RISC core cycle |
| P — Program word fetch: always an aligned word read, lasts for at least one RISC core cycle |
| r — 8 bit data read: lasts for at least one RISC core cycle |
| R — 16 bit data read: lasts for at least one RISC core cycle |
| w — 8 bit data write: lasts for at least one RISC core cycle |
| W — 16 bit data write: lasts for at least one RISC core cycle |
| A — Alignment cycle: no read or write, lasts for zero or one RISC core cycles |
| f — Free cycle: no read or write, lasts for one RISC core cycles |
| **Special Cases** |
| PP/P — Branch: PP if branch taken, P if not |

## 10.8.4 Thread Execution

When the RISC core is triggered by an interrupt request (see Figure 10-1) it first executes a vector fetch sequence which performs three bus accesses:

1. A V-cycle to fetch the initial content of the program counter.
2. A V-cycle to fetch the initial content of the data segment pointer (R1).
3. A P-cycle to load the initial opcode.

Afterwards a sequence of instructions (thread) is executed which is terminated by an "RTS" instruction. If further interrupt requests are pending after a thread has been terminated, a new vector fetch will be performed. Otherwise the RISC core will either resume a previous thread (beginning with a P-cycle to refetch the interrupted opcode) or it will become idle until a new interrupt request is received. A thread can only be interrupted by an interrupt request of higher priority.

## 10.8.5 Instruction Glossary

This section describes the XGATE instruction set in alphabetical order.

# ADC — Add with Carry — ADC

## Operation

RS1 + RS2 + C ⇒ RD

Adds the content of register RS1, the content of register RS2 and the value of the Carry bit using binary addition and stores the result in the destination register RD. The Zero Flag is also carried forward from the previous operation allowing 32 and more bit additions.

Example:

```
ADD     R6,R2,R2
ADC     R7,R3,R3 ; R7:R6 = R5:R4 + R3:R2
BCC              ; conditional branch on 32 bit addition
```

## CCR Effects

| N | Z | V | C |
|---|---|---|---|
| Δ | Δ | Δ | Δ |

N: Set if bit 15 of the result is set; cleared otherwise.

Z: Set if the result is $0000 and Z was set before this operation; cleared otherwise.

V: Set if a two´s complement overflow resulted from the operation; cleared otherwise.
RS1[15] & RS2[15] & $\overline{RD[15]}_{new}$ | $\overline{RS1[15]}$ & $\overline{RS2[15]}$ & $RD[15]_{new}$

C: Set if there is a carry from bit 15 of the result; cleared otherwise.
RS1[15] & RS2[15] | RS1[15] & $\overline{RD[15]}_{new}$ | RS2[15] & $\overline{RD[15]}_{new}$

## Code and CPU Cycles

| Source Form | Address Mode | Machine Code | | | | | | | | | | Cycles |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| ADC RD, RS1, RS2 | TRI | 0 | 0 | 0 | 1 | 1 | RD | RS1 | RS2 | 1 | 1 | P |

# ADD — Add without Carry — ADD

## Operation

RS1 + RS2 ⇒ RD
RD + IMM16 ⇒ RD (translates to ADDL RD, #IMM16[7:0]; ADDH RD, #IMM16[15:8])

Performs a 16 bit addition and stores the result in the destination register RD.

**NOTE**

When using immediate addressing mode (ADD RD, #IMM16), the V-flag and the C-Flag of the first instruction (ADDL RD, #IMM16[7:0]) are not considered by the second instruction (ADDH RD, #IMM16[15:8]).
⇒ Don't rely on the V-Flag if RD + IMM16[7:0] ≥ $2^{15}$.
⇒ Don't rely on the C-Flag if RD + IMM16[7:0] ≥ $2^{16}$.

## CCR Effects

| N | Z | V | C |
|---|---|---|---|
| Δ | Δ | Δ | Δ |

N: Set if bit 15 of the result is set; cleared otherwise.

Z: Set if the result is $0000; cleared otherwise.

V: Set if a two´s complement overflow resulted from the operation; cleared otherwise.
RS1[15] & RS2[15] & $\overline{RD[15]}_{new}$ | $\overline{RS1[15]}$ & $\overline{RS2[15]}$ & $RD[15]_{new}$
Refer to ADDH instruction for #IMM16 operations.

C: Set if there is a carry from bit 15 of the result; cleared otherwise.
RS1[15] & RS2[15] | RS1[15] & $\overline{RD[15]}_{new}$ | RS2[15] & $\overline{RD[15]}_{new}$
Refer to ADDH instruction for #IMM16 operations.

## Code and CPU Cycles

| Source Form | Address Mode | Machine Code | | | | | | | | | | Cycles |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| ADD RD, RS1, RS2 | TRI | 0 | 0 | 0 | 1 | 1 | RD | RS1 | RS2 | 1 | 0 | P |
| ADD RD, #IMM16 | IMM8 | 1 | 1 | 1 | 0 | 0 | RD | IMM16[7:0] | | | | P |
| | IMM8 | 1 | 1 | 1 | 0 | 1 | RD | IMM16[15:8] | | | | P |

# ADDH — Add Immediate 8 bit Constant (High Byte) — ADDH

## Operation

RD + IMM8:$00 ⇒ RD

Adds the content of high byte of register RD and a signed immediate 8 bit constant using binary addition and stores the result in the high byte of the destination register RD. This instruction can be used after an ADDL for a 16 bit immediate addition.

Example:

```
ADDL    R2,#LOWBYTE
ADDH    R2,#HIGHBYTE      ; R2 = R2 + 16 bit immediate
```

## CCR Effects

| N | Z | V | C |
|---|---|---|---|
| Δ | Δ | Δ | Δ |

N: Set if bit 15 of the result is set; cleared otherwise.

Z: Set if the result is $0000; cleared otherwise.

V: Set if a two´s complement overflow resulted from the operation; cleared otherwise.
$RD[15]_{old}$ & IMM8[7] & $\overline{RD[15]}_{new}$ | $\overline{RD[15]}_{old}$ & $\overline{IMM8[7]}$ & $RD[15]_{new}$

C: Set if there is a carry from the bit 15 of the result; cleared otherwise.
$RD[15]_{old}$ & IMM8[7] | $RD[15]_{old}$ & $\overline{RD[15]}_{new}$ | IMM8[7] & $\overline{RD[15]}_{new}$

## Code and CPU Cycles

| Source Form | Address Mode | Machine Code | | | | | | | Cycles |
|---|---|---|---|---|---|---|---|---|---|
| ADDH RD, #IMM8 | IMM8 | 1 | 1 | 1 | 0 | 1 | RD | IMM8 | P |

# ADDL — Add Immediate 8 bit Constant (Low Byte) — ADDL

## Operation

RD + $00:IMM8 ⇒ RD

Adds the content of register RD and an unsigned immediate 8 bit constant using binary addition and stores the result in the destination register RD. This instruction must be used first for a 16 bit immediate addition in conjunction with the ADDH instruction.

## CCR Effects

| N | Z | V | C |
|---|---|---|---|
| Δ | Δ | Δ | Δ |

N: Set if bit 15 of the result is set; cleared otherwise.

Z: Set if the result is $0000; cleared otherwise.

V: Set if a two´s complement overflow resulted from the 8 bit operation; cleared otherwise.
$\overline{RD[15]}_{old}$ & $RD[15]_{new}$

C: Set if there is a carry from the bit 15 of the result; cleared otherwise.
$RD[15]_{old}$ & $\overline{RD[15]}_{new}$

## Code and CPU Cycles

| Source Form | Address Mode | Machine Code | | | | | | | Cycles |
|---|---|---|---|---|---|---|---|---|---|
| ADDL RD, #IMM8 | IMM8 | 1 | 1 | 1 | 0 | 0 | RD | IMM8 | P |

# AND — Logical AND — AND

## Operation

RS1 & RS2 ⇒ RD
RD & IMM16 ⇒ RD (translates to ANDL RD, #IMM16[7:0]; ANDH RD, #IMM16[15:8])

Performs a bit wise logical AND of two 16 bit values and stores the result in the destination register RD.

**NOTE**

When using immediate addressing mode (AND RD, #IMM16), the Z-flag of the first instruction (ANDL RD, #IMM16[7:0]) is not considered by the second instruction (ANDH RD, #IMM16[15:8]).
⇒ Don't rely on the Z-Flag.

## CCR Effects

| N | Z | V | C |
|---|---|---|---|
| Δ | Δ | 0 | — |

N: Set if bit 15 of the result is set; cleared otherwise.
Z: Set if the result is $0000; cleared otherwise.
Refer to ANDH instruction for #IMM16 operations.
V: 0; cleared.
C: Not affected.

## Code and CPU Cycles

| Source Form | Address Mode | Machine Code | | | | | | | | | | Cycles |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| AND RD, RS1, RS2 | TRI | 0 | 0 | 0 | 1 | 0 | RD | RS1 | RS2 | 0 | 0 | P |
| AND RD, #IMM16 | IMM8 | 1 | 0 | 0 | 0 | 0 | RD | IMM16[7:0] | | | | P |
| | IMM8 | 1 | 0 | 0 | 0 | 1 | RD | IMM16[15:8] | | | | P |

# ANDH — Logical AND Immediate 8 bit Constant (High Byte) — ANDH

## Operation

RD.H & IMM8 ⇒ RD.H

Performs a bit wise logical AND between the high byte of register RD and an immediate 8 bit constant and stores the result in the destination register RD.H. The low byte of RD is not affected.

## CCR Effects

| N | Z | V | C |
|---|---|---|---|
| Δ | Δ | 0 | — |

N: Set if bit 15 of the result is set; cleared otherwise.
Z: Set if the 8 bit result is $00; cleared otherwise.
V: 0; cleared.
C: Not affected.

## Code and CPU Cycles

| Source Form | Address Mode | Machine Code | | | | | | | Cycles |
|---|---|---|---|---|---|---|---|---|---|
| ANDH RD, #IMM8 | IMM8 | 1 | 0 | 0 | 0 | 1 | RD | IMM8 | P |

# ANDL — Logical AND Immediate 8 bit Constant (Low Byte) — ANDL

## Operation

RD.L & IMM8 ⇒ RD.L

Performs a bit wise logical AND between the low byte of register RD and an immediate 8 bit constant and stores the result in the destination register RD.L. The high byte of RD is not affected.

## CCR Effects

| N | Z | V | C |
|---|---|---|---|
| Δ | Δ | 0 | — |

N: Set if bit 7 of the result is set; cleared otherwise.
Z: Set if the 8 bit result is $00; cleared otherwise.
V: 0; cleared.
C: Not affected.

## Code and CPU Cycles

| Source Form | Address Mode | Machine Code | | | | | | | Cycles |
|---|---|---|---|---|---|---|---|---|---|
| ANDL RD, #IMM8 | IMM8 | 1 | 0 | 0 | 0 | 0 | RD | IMM8 | P |

# ASR — Arithmetic Shift Right — ASR

## Operation

*n* = RS or IMM4

Shifts the bits in register RD *n* positions to the right. The higher *n* bits of the register RD become filled with the sign bit (RD[15]). The carry flag will be updated to the bit contained in RD[n-1] before the shift for $n > 0$.

*n* can range from 0 to 16.

In immediate address mode, *n* is determined by the operand IMM4. *n* is considered to be 16 if IMM4 is equal to 0.

In dyadic address mode, *n* is determined by the content of RS. *n* is considered to be 16 if the content of RS is greater than 15.

## CCR Effects

| N | Z | V | C |
|---|---|---|---|
| Δ | Δ | Δ | Δ |

N: Set if bit 15 of the result is set; cleared otherwise.

Z: Set if the result is $0000; cleared otherwise.

V: Set if a two´s complement overflow resulted from the operation; cleared otherwise.
$RD[15]_{old}$ ^ $RD[15]_{new}$

C: Set if $n > 0$ and RD[n-1] = 1; if n = 0 unaffected.

## Code and CPU Cycles

| Source Form | Address Mode | Machine Code | | | | | | | | | | | | Cycles |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| ASR RD, #IMM4 | IMM4 | 0 | 0 | 0 | 0 | 1 | RD | IMM4 | | 1 | 0 | 0 | 1 | P |
| ASR RD, RS | DYA | 0 | 0 | 0 | 0 | 1 | RD | RS | 1 | 0 | 0 | 0 | 1 | P |

# BCC

## Branch if Carry Cleared (Same as BHS)

BCC

### Operation

If C = 0, then PC + $0002 + (REL9 << 1) ⇒ PC

Tests the Carry flag and branches if C = 0.

### CCR Effects

| N | Z | V | C |
|---|---|---|---|
| — | — | — | — |

N: Not affected.
Z: Not affected.
V: Not affected.
C: Not affected.

### Code and CPU Cycles

| Source Form | Address Mode | Machine Code | | | | | | | | Cycles |
|---|---|---|---|---|---|---|---|---|---|---|
| BCC REL9 | REL9 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | REL9 | PP/P |

# BCS

## Branch if Carry Set (Same as BLO)

BCS

### Operation

If C = 1, then PC + $0002 + (REL9 << 1) ⇒ PC

Tests the Carry flag and branches if C = 1.

### CCR Effects

| N | Z | V | C |
|---|---|---|---|
| — | — | — | — |

N: Not affected.
Z: Not affected.
V: Not affected.
C: Not affected.

### Code and CPU Cycles

| Source Form | Address Mode | Machine Code | | | | | | | | Cycles |
|---|---|---|---|---|---|---|---|---|---|---|
| BCS REL9 | REL9 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | REL9 | PP/P |

# BEQ Branch if Equal BEQ

## Operation

If Z = 1, then PC + $0002 + (REL9 << 1) ⇒ PC

Tests the Zero flag and branches if Z = 1.

## CCR Effect

| N | Z | V | C |
|---|---|---|---|
| — | — | — | — |

N: Not affected.
Z: Not affected.
V: Not affected.
C: Not affected.

## Code and CPU Cycles

| Source Form | Address Mode | Machine Code | | | | | | | | Cycles |
|---|---|---|---|---|---|---|---|---|---|---|
| BEQ REL9 | REL9 | 0 | 0 | 1 | 0 | 0 | 1 | 1 | REL9 | PP/P |

# BFEXT Bit Field Extract BFEXT

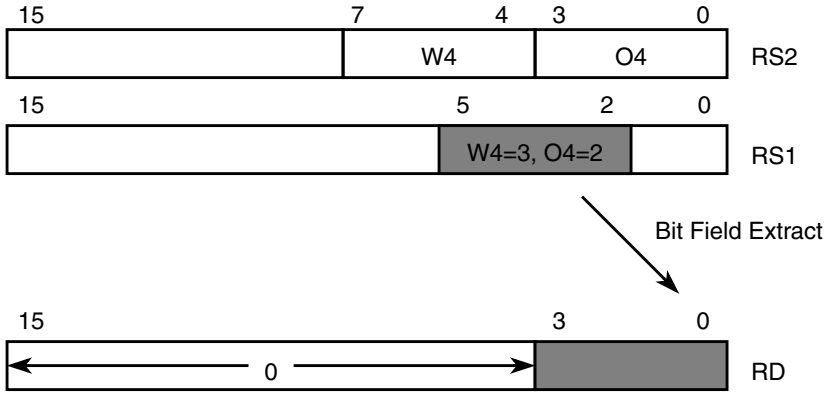
## Operation

RS1[(*o*+*w*):*o*] ⇒ RD[*w*:0]; 0 ⇒ RD[15:(*w*+1)]

*w* = (RS2[7:4])

*o* = (RS2[3:0])

Extracts *w+1* bits from register RS1 starting at position *o* and writes them right aligned into register RD. The remaining bits in RD will be cleared. If (o+w) > 15 only bits [15:o] get extracted.




## CCR Effects

| N | Z | V | C |
|---|---|---|---|
| Δ | Δ | 0 | — |

N: Set if bit 15 of the result is set; cleared otherwise.

Z: Set if the result is $0000; cleared otherwise.

V: 0; cleared.

C: Not affected.

## Code and CPU Cycles

| Source Form | Address Mode | Machine Code | | | | | | | | | | Cycles |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| BFEXT RD, RS1, RS2 | TRI | 0 | 1 | 1 | 0 | 0 | RD | RS1 | RS2 | 1 | 1 | P |

# BFFO Bit Field Find First One BFFO

## Operation

FirstOne(RS) ⇒ RD;

Searches the first “1” in register RS (from MSB to LSB) and writes the bit position into the destination register RD. The upper bits of RD are cleared. In case the content of RS is equal to $0000, RD will be cleared and the carry flag will be set. This is used to distinguish a “1” in position 0 versus no “1” in the whole RS register at all.

## CCR Effects

| N | Z | V | C |
|---|---|---|---|
| 0 | Δ | 0 | Δ |

N: 0; cleared.

Z: Set if the result is $0000; cleared otherwise.

V: 0; cleared.

C: Set if RS = $0000(1); cleared otherwise.

1. Before executing the instruction

## Code and CPU Cycles

| Source Form | Address Mode | Machine Code | | | | | | | | | | | | Cycles |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| BFFO RD, RS | DYA | 0 | 0 | 0 | 0 | 1 | RD | RS | 1 | 0 | 0 | 0 | 0 | P |

# BFINS Bit Field Insert BFINS

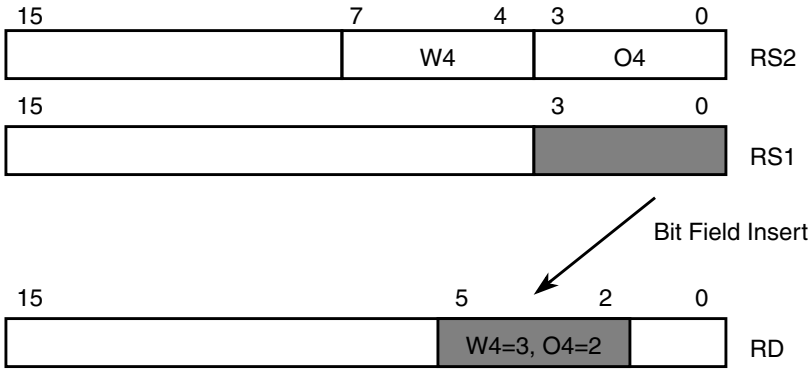
## Operation

RS1[*w*:*0*] ⇒ RD[(*w*+*o*):o];

*w* = (RS2[7:4])

*o* = (RS2[3:0])

Extracts *w+1* bits from register RS1 starting at position 0 and writes them into register RD starting at position *o*. The remaining bits in RD are not affected. If (o+w) > 15 the upper bits are ignored. Using R0 as a RS1, this command can be used to clear bits.




## CCR Effects

| N | Z | V | C |
|---|---|---|---|
| Δ | Δ | 0 | — |

N: Set if bit 15 of the result is set; cleared otherwise.

Z: Set if the result is $0000; cleared otherwise.

V: 0; cleared.

C: Not affected.

## Code and CPU Cycles

| Source Form | Address Mode | Machine Code | | | | | | | | | | Cycles |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| BFINS RD, RS1, RS2 | TRI | 0 | 1 | 1 | 0 | 1 | RD | RS1 | RS2 | 1 | 1 | P |

# BFINSI Bit Field Insert and Invert BFINSI

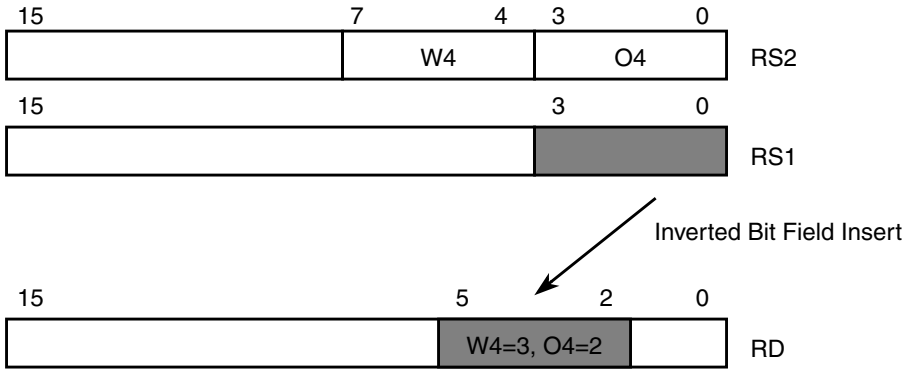
## Operation

!RS1[*w*:*0*] ⇒ RD[*w*+o:o];

*w* = (RS2[7:4])

*o* = (RS2[3:0])

Extracts *w+1* bits from register RS1 starting at position 0, inverts them and writes into register RD starting at position *o*. The remaining bits in RD are not affected. If (o+w) > 15 the upper bits are ignored. Using R0 as a RS1, this command can be used to set bits.




## CCR Effects

| N | Z | V | C |
|---|---|---|---|
| Δ | Δ | 0 | — |

N: Set if bit 15 of the result is set; cleared otherwise.

Z: Set if the result is $0000; cleared otherwise.

V: 0; cleared.

C: Not affected.

## Code and CPU Cycles

| Source Form | Address Mode | Machine Code | | | | | | | | | | Cycles |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| BFINSI RD, RS1, RS2 | TRI | 0 | 1 | 1 | 1 | 0 | RD | RS1 | RS2 | 1 | 1 | P |

# BFINSX Bit Field Insert and XNOR BFINSX

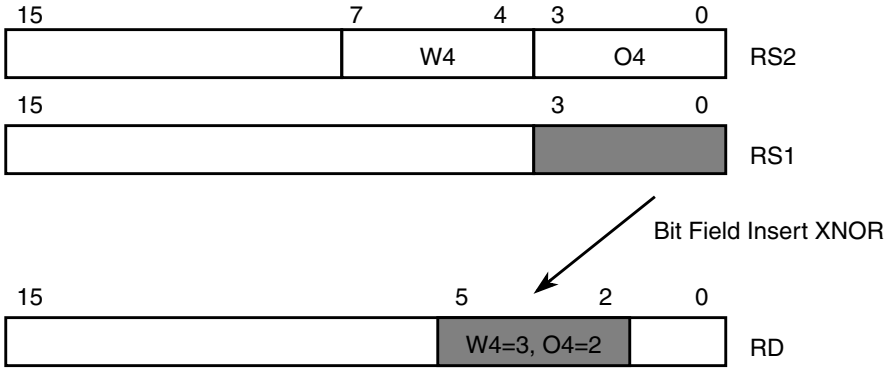
## Operation

!(RS1[*w*:*0*] ^ RD[w+o:o]) ⇒ RD[*w*+o:o];

*w* = (RS2[7:4])

*o* = (RS2[3:0])

Extracts *w+1* bits from register RS1 starting at position 0, performs an XNOR with RD[w+o:o] and writes the bits back to RD. The remaining bits in RD are not affected. If (o+w) > 15 the upper bits are ignored. Using R0 as a RS1, this command can be used to toggle bits.




## CCR Effects

| N | Z | V | C |
|---|---|---|---|
| Δ | Δ | 0 | — |

N: Set if bit 15 of the result is set; cleared otherwise.

Z: Set if the result is $0000; cleared otherwise.

V: 0; cleared.

C: Not affected.

## Code and CPU Cycles

| Source Form | Address Mode | Machine Code | | | | | | | | | | Cycles |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| BFINSX RD, RS1, RS2 | TRI | 0 | 1 | 1 | 1 | 1 | RD | RS1 | RS2 | 1 | 1 | P |

# BGE Branch if Greater than or Equal to Zero BGE

## Operation

If N ^ V = 0, then PC + $0002 + (REL9 << 1) ⇒ PC

Branch instruction to compare signed numbers.

Branch if RS1 ≥ RS2:

```
SUB     R0,RS1,RS2
BGE     REL9
```

## CCR Effects

| N | Z | V | C |
|---|---|---|---|
| — | — | — | — |

N: Not affected.
Z: Not affected.
V: Not affected.
C: Not affected.

## Code and CPU Cycles

| Source Form | Address Mode | Machine Code | | | | | | | | Cycles |
|---|---|---|---|---|---|---|---|---|---|---|
| BGE REL9 | REL9 | 0 | 0 | 1 | 1 | 0 | 1 | 0 | REL9 | PP/P |

# BGT Branch if Greater than Zero BGT

## Operation

If Z | (N ^ V) = 0, then PC + $0002 + (REL9 << 1) ⇒ PC

Branch instruction to compare signed numbers.

Branch if RS1 > RS2:

```
SUB     R0,RS1,RS2
BGT     REL9
```

## CCR Effects

| N | Z | V | C |
|---|---|---|---|
| — | — | — | — |

N: Not affected.
Z: Not affected.
V: Not affected.
C: Not affected.

## Code and CPU Cycles

| Source Form | Address Mode | Machine Code | | | | | | | | Cycles |
|---|---|---|---|---|---|---|---|---|---|---|
| BGT REL9 | REL9 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | REL9 | PP/P |

# BHI Branch if Higher BHI

## Operation

If C | Z = 0, then PC + $0002 + (REL9 << 1) ⇒ PC

Branch instruction to compare unsigned numbers.

Branch if RS1 > RS2:

```
SUB     R0,RS1,RS2
BHI     REL9
```

## CCR Effects

| N | Z | V | C |
|---|---|---|---|
| — | — | — | — |

N: Not affected.
Z: Not affected.
V: Not affected.
C: Not affected.

## Code and CPU Cycles

| Source Form | Address Mode | Machine Code | | | | | | | | Cycles |
|---|---|---|---|---|---|---|---|---|---|---|
| BHI REL9 | REL9 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | REL9 | PP/P |

# BHS

## Branch if Higher or Same (Same as BCC)

BHS

### Operation

If C = 0, then PC + $0002 + (REL9 << 1) ⇒ PC

Branch instruction to compare unsigned numbers.

Branch if RS1 ≥ RS2:

```
SUB     R0,RS1,RS2
BHS     REL9
```

### CCR Effects

| N | Z | V | C |
|---|---|---|---|
| — | — | — | — |

N: Not affected.
Z: Not affected.
V: Not affected.
C: Not affected.

### Code and CPU Cycles

| Source Form | Address Mode | Machine Code | | | | | | | | Cycles |
|---|---|---|---|---|---|---|---|---|---|---|
| BHS REL9 | REL9 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | REL9 | PP/P |

# BITH — Bit Test Immediate 8 bit Constant (High Byte) — BITH

## Operation

RD.H & IMM8 ⇒ NONE

Performs a bit wise logical AND between the high byte of register RD and an immediate 8 bit constant. Only the condition code flags get updated, but no result is written back.

## CCR Effects

| N | Z | V | C |
|---|---|---|---|
| Δ | Δ | 0 | — |

N: Set if bit 15 of the result is set; cleared otherwise.
Z: Set if the 8 bit result is $00; cleared otherwise.
V: 0; cleared.
C: Not affected.

## Code and CPU Cycles

| Source Form | Address Mode | Machine Code | | | | | | | Cycles |
|---|---|---|---|---|---|---|---|---|---|
| BITH RD, #IMM8 | IMM8 | 1 | 0 | 0 | 1 | 1 | RD | IMM8 | P |

# BITL

## Bit Test Immediate 8 bit Constant (Low Byte)

BITL

### Operation

RD.L & IMM8 ⇒ NONE

Performs a bit wise logical AND between the low byte of register RD and an immediate 8 bit constant. Only the condition code flags get updated, but no result is written back.

### CCR Effects

| N | Z | V | C |
|---|---|---|---|
| Δ | Δ | 0 | — |

N: Set if bit 7 of the result is set; cleared otherwise.
Z: Set if the 8 bit result is $00; cleared otherwise.
V: 0; cleared.
C: Not affected.

### Code and CPU Cycles

| Source Form | Address Mode | Machine Code | | | | | | | Cycles |
|---|---|---|---|---|---|---|---|---|---|
| BITL RD, #IMM8 | IMM8 | 1 | 0 | 0 | 1 | 0 | RD | IMM8 | P |

# BLE Branch if Less or Equal to Zero BLE

## Operation

If Z | (N ^ V) = 1, then PC + $0002 + (REL9 << 1) ⇒ PC

Branch instruction to compare signed numbers.

Branch if RS1 ≤ RS2:

```
SUB     R0,RS1,RS2
BLE     REL9
```

## CCR Effects

| N | Z | V | C |
|---|---|---|---|
| — | — | — | — |

N: Not affected.
Z: Not affected.
V: Not affected.
C: Not affected.

## Code and CPU Cycles

| Source Form | Address Mode | Machine Code | | | | | | | | Cycles |
|---|---|---|---|---|---|---|---|---|---|---|
| BLE REL9 | REL9 | 0 | 0 | 1 | 1 | 1 | 0 | 1 | REL9 | PP/P |

# BLO

## Branch if Carry Set (Same as BCS)

BLO

### Operation

If C = 1, then PC + $0002 + (REL9 << 1) ⇒ PC

Branch instruction to compare unsigned numbers.

Branch if RS1 < RS2:

```
SUB     R0,RS1,RS2
BLO     REL9
```

### CCR Effects

| N | Z | V | C |
|---|---|---|---|
| — | — | — | — |

N: Not affected.
Z: Not affected.
V: Not affected.
C: Not affected.

### Code and CPU Cycles

| Source Form | Address Mode | Machine Code | | | | | | | | Cycles |
|---|---|---|---|---|---|---|---|---|---|---|
| BLO REL9 | REL9 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | REL9 | PP/P |

# BLS Branch if Lower or Same BLS

## Operation

If C | Z = 1, then PC + $0002 + (REL9 << 1) ⇒ PC

Branch instruction to compare unsigned numbers.

Branch if RS1 ≤ RS2:

```
SUB     R0,RS1,RS2
BLS     REL9
```

## CCR Effects

| N | Z | V | C |
|---|---|---|---|
| — | — | — | — |

N: Not affected.
Z: Not affected.
V: Not affected.
C: Not affected.

## Code and CPU Cycles

| Source Form | Address Mode | Machine Code | | | | | | | | Cycles |
|---|---|---|---|---|---|---|---|---|---|---|
| BLS REL9 | REL9 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | REL9 | PP/P |

# BLT Branch if Lower than Zero BLT

## Operation

If N ^ V = 1, then PC + $0002 + (REL9 << 1) ⇒ PC

Branch instruction to compare signed numbers.

Branch if RS1 < RS2:

```
SUB     R0,RS1,RS2
BLT     REL9
```

## CCR Effects

| N | Z | V | C |
|---|---|---|---|
| — | — | — | — |

N: Not affected.
Z: Not affected.
V: Not affected.
C: Not affected.

## Code and CPU Cycles

| Source Form | Address Mode | Machine Code | | | | | | | | Cycles |
|---|---|---|---|---|---|---|---|---|---|---|
| BLT REL9 | REL9 | 0 | 0 | 1 | 1 | 0 | 1 | 1 | REL9 | PP/P |

# BMI Branch if Minus BMI

## Operation

If N = 1, then PC + $0002 + (REL9 << 1) ⇒ PC

Tests the sign flag and branches if N = 1.

## CCR Effects

| N | Z | V | C |
|---|---|---|---|
| — | — | — | — |

N: Not affected.
Z: Not affected.
V: Not affected.
C: Not affected.

## Code and CPU Cycles

| Source Form | Address Mode | Machine Code | | | | | | | | Cycles |
|---|---|---|---|---|---|---|---|---|---|---|
| BMI REL9 | REL9 | 0 | 0 | 1 | 0 | 1 | 0 | 1 | REL9 | PP/P |

# BNE Branch if Not Equal BNE

## Operation

If Z = 0, then PC + $0002 + (REL9 << 1) ⇒ PC

Tests the Zero flag and branches if Z = 0.

## CCR Effects

| N | Z | V | C |
|---|---|---|---|
| — | — | — | — |

N: Not affected.
Z: Not affected.
V: Not affected.
C: Not affected.

## Code and CPU Cycles

| Source Form | Address Mode | Machine Code | | | | | | | | Cycles |
|---|---|---|---|---|---|---|---|---|---|---|
| BNE REL9 | REL9 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | REL9 | PP/P |

# BPL Branch if Plus BPL

## Operation

If N = 0, then PC + $0002 + (REL9 << 1) ⇒ PC

Tests the Sign flag and branches if N = 0.

## CCR Effects

| N | Z | V | C |
|---|---|---|---|
| — | — | — | — |

N: Not affected.
Z: Not affected.
V: Not affected.
C: Not affected.

## Code and CPU Cycles

| Source Form | Address Mode | Machine Code | | | | | | | | Cycles |
|---|---|---|---|---|---|---|---|---|---|---|
| BPL REL9 | REL9 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | REL9 | PP/P |

# BRA Branch Always BRA

## Operation

PC + $0002 + (REL10 << 1) ⇒ PC

Branches always.

## CCR Effects

| N | Z | V | C |
|---|---|---|---|
| — | — | — | — |

N: Not affected.
Z: Not affected.
V: Not affected.
C: Not affected.

## Code and CPU Cycles

| Source Form | Address Mode | Machine Code | | | | | | | Cycles |
|---|---|---|---|---|---|---|---|---|---|
| BRA REL10 | REL10 | 0 | 0 | 1 | 1 | 1 | 1 | REL10 | PP |

# BRK Break BRK

## Operation

Put XGATE into Debug Mode (see Section 10.6.1.0.1, “Entering Debug Mode”) and signals a software breakpoint to the S12X_DBG module (see section 4.9 of the **S12X_DBG Section**).

**NOTE**

It is not possible to single step over a BRK instruction. This instruction does not advance the program counter.

## CCR Effects

| N | Z | V | C |
|---|---|---|---|
| — | — | — | — |

N: Not affected.
Z: Not affected.
V: Not affected.
C: Not affected.

## Code and CPU Cycles

| Source Form | Address Mode | Machine Code | | | | | | | | | | | | | | | | Cycles |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| BRK | INH | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | PAff |

# BVC Branch if Overflow Cleared BVC

## Operation

If V = 0, then PC + $0002 + (REL9 << 1) ⇒ PC

Tests the Overflow flag and branches if V = 0.

## CCR Effects

| N | Z | V | C |
|---|---|---|---|
| — | — | — | — |

N: Not affected.
Z: Not affected.
V: Not affected.
C: Not affected.

## Code and CPU Cycles

| Source Form | Address Mode | Machine Code | | | | | | | | Cycles |
|---|---|---|---|---|---|---|---|---|---|---|
| BVC REL9 | REL9 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | REL9 | PP/P |

# BVS Branch if Overflow Set BVS

## Operation

If V = 1, then PC + $0002 + (REL9 << 1) ⇒ PC

Tests the Overflow flag and branches if V = 1.

## CCR Effects

| N | Z | V | C |
|---|---|---|---|
| — | — | — | — |

N: Not affected.
Z: Not affected.
V: Not affected.
C: Not affected.

## Code and CPU Cycles

| Source Form | Address Mode | Machine Code | | | | | | | | Cycles |
|---|---|---|---|---|---|---|---|---|---|---|
| BVS REL9 | REL9 | 0 | 0 | 1 | 0 | 1 | 1 | 1 | REL9 | PP/P |

# CMP — Compare — CMP

## Operation

RS1 – RS2 ⇒ NONE (translates to SUB R0, RS1, RS2)
RD – IMM16 ⇒ NONE (translates to CMPL RD, #IMM16[7:0]; CPCH RD, #IMM16[15:8])

Subtracts two 16 bit values and discards the result.

## CCR Effects

| N | Z | V | C |
|---|---|---|---|
| Δ | Δ | Δ | Δ |

N: Set if bit 15 of the result is set; cleared otherwise.

Z: Set if the result is $0000; cleared otherwise.

V: Set if a two´s complement overflow resulted from the operation; cleared otherwise.
RS1[15] & $\overline{RS2[15]}$ & $\overline{result[15]}$ | $\overline{RS1[15]}$ & RS2[15] & result[15]
RD[15] & $\overline{IMM16[15]}$ & $\overline{result[15]}$ | $\overline{RD[15]}$ & IMM16[15] & result[15]

C: Set if there is a carry from the bit 15 of the result; cleared otherwise.
$\overline{RS1[15]}$ & RS2[15] | $\overline{RS1[15]}$ & result[15] | RS2[15] & result[15]
$\overline{RD[15]}$ & IMM16[15] | $\overline{RD[15]}$ & result[15] | IMM16[15] & result[15]

## Code and CPU Cycles

| Source Form | Address Mode | Machine Code | | | | | | | | | | | Cycles |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| CMP RS1, RS2 | TRI | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | RS1 | RS2 | 0 | 0 | P |
| CMP RS, #IMM16 | IMM8 | 1 | 1 | 0 | 1 | 0 | RS | | | IMM16[7:0] | | | | P |
| | IMM8 | 1 | 1 | 0 | 1 | 1 | RS | | | IMM16[15:8] | | | | P |

# CMPL — Compare Immediate 8 bit Constant (Low Byte) — CMPL

## Operation

RS.L – IMM8 ⇒ NONE, only condition code flags get updated

Subtracts the 8 bit constant IMM8 contained in the instruction code from the low byte of the source register RS.L using binary subtraction and updates the condition code register accordingly.

Remark: There is no equivalent operation using triadic addressing. Comparing the values of two registers can be performed by using the subtract instruction with R0 as destination register.

## CCR Effects

| N | Z | V | C |
|---|---|---|---|
| Δ | Δ | Δ | Δ |

N: Set if bit 7 of the result is set; cleared otherwise.

Z: Set if the 8 bit result is $00; cleared otherwise.

V: Set if a two´s complement overflow resulted from the 8 bit operation; cleared otherwise.
RS[7] & $\overline{\text{IMM8[7]}}$ & $\overline{\text{result[7]}}$ | $\overline{\text{RS[7]}}$ & IMM8[7] & result[7]

C: Set if there is a carry from the Bit 7 to Bit 8 of the result; cleared otherwise.
$\overline{\text{RS[7]}}$ & IMM8[7] | $\overline{\text{RS[7]}}$ & result[7] | IMM8[7] & result[7]

## Code and CPU Cycles

| Source Form | Address Mode | Machine Code | | | | | | | Cycles |
|---|---|---|---|---|---|---|---|---|---|
| CMPL RS, #IMM8 | IMM8 | 1 | 1 | 0 | 1 | 0 | RS | IMM8 | P |

# COM One's Complement COM

## Operation

~RS ⇒ RD (translates to XNOR RD, R0, RS)
~RD ⇒ RD (translates to XNOR RD, R0, RD)

Performs a one's complement on a general purpose register.

## CCR Effects

| N | Z | V | C |
|---|---|---|---|
| Δ | Δ | 0 | — |

N: Set if bit 15 of the result is set; cleared otherwise.
Z: Set if the result is $0000; cleared otherwise.
V: 0; cleared.
C: Not affected.

## Code and CPU Cycles

| Source Form | Address Mode | Machine Code | | | | | | | | | | Cycles |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| COM RD, RS | TRI | 0 | 0 | 0 | 1 | 0 | RD | 0 | 0 | 0 | RS | 1 | 1 | P |
| COM RD | TRI | 0 | 0 | 0 | 1 | 0 | RD | 0 | 0 | 0 | RD | 1 | 1 | P |

# CPC Compare with Carry CPC

## Operation

RS1 – RS2 - C ⇒ NONE (translates to SBC R0, RS1, RS2)

Subtracts the carry bit and the content of register RS2 from the content of register RS1 using binary subtraction and discards the result.

## CCR Effects

| N | Z | V | C |
|---|---|---|---|
| Δ | Δ | Δ | Δ |

N: Set if bit 15 of the result is set; cleared otherwise.

Z: Set if the result is $0000; cleared otherwise.

V: Set if a two´s complement overflow resulted from the operation; cleared otherwise.
RS1[15] & $\overline{RS2[15]}$ & $\overline{result[15]}$ | $\overline{RS1[15]}$ & RS2[15] & result[15]

C: Set if there is a carry from the bit 15 of the result; cleared otherwise.
$\overline{RS1[15]}$ & RS2[15] | $\overline{RS1[15]}$ & result[15] | RS2[15] & result[15]

## Code and CPU Cycles

| Source Form | Address Mode | Machine Code | | | | | | | | | | | Cycles |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| CPC RS1, RS2 | TRI | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | RS1 | RS2 | 0 | 1 | P |

# CPCH — Compare Immediate 8 bit Constant with Carry (High Byte) — CPCH

## Operation

RS.H - IMM8 - C ⇒ NONE, only condition code flags get updated

Subtracts the carry bit and the 8 bit constant IMM8 contained in the instruction code from the high byte of the source register RD using binary subtraction and updates the condition code register accordingly. The carry bit and Zero bits are taken into account to allow a 16 bit compare in the form of

```
CMPL    R2,#LOWBYTE
CPCH    R2,#HIGHBYTE
BCC                     ; branch condition
```

Remark: There is no equivalent operation using triadic addressing. Comparing the values of two registers can be performed by using the subtract instruction with R0 as destination register.

## CCR Effects

| N | Z | V | C |
|---|---|---|---|
| Δ | Δ | Δ | Δ |

N: Set if bit 15 of the result is set; cleared otherwise.

Z: Set if the result is $00 and Z was set before this operation; cleared otherwise.

V: Set if a two´s complement overflow resulted from the operation; cleared otherwise.
RS[15] & $\overline{\text{IMM8[7]}}$ & $\overline{\text{result[15]}}$ | $\overline{\text{RS[15]}}$ & IMM8[7] & result[15]

C: Set if there is a carry from the bit 15 of the result; cleared otherwise.
$\overline{\text{RS[15]}}$ & IMM8[7] | $\overline{\text{RS[15]}}$ & result[15] | IMM8[7] & result[15]

## Code and CPU Cycles

| Source Form | Address Mode | Machine Code | | | | | | | Cycles |
|---|---|---|---|---|---|---|---|---|---|
| CPCH RD, #IMM8 | IMM8 | 1 | 1 | 0 | 1 | 1 | RS | IMM8 | P |

# CSEM Clear Semaphore CSEM

## Operation

Unlocks a semaphore that was locked by the RISC core.

In monadic address mode, bits RS[2:0] select the semaphore to be cleared.

## CCR Effects

| N | Z | V | C |
|---|---|---|---|
| — | — | — | — |

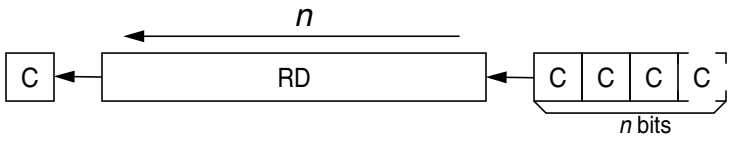
N: Not affected.
Z: Not affected.
V: Not affected.
C: Not affected.

## Code and CPU Cycles

| Source Form | Address Mode | Machine Code | | | | | | | | | | | | | | Cycles |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| CSEM #IMM3 | IMM3 | 0 | 0 | 0 | 0 | 0 | IMM3 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | PA |
| CSEM RS | MON | 0 | 0 | 0 | 0 | 0 | RS | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 1 | PA |

# CSL — Logical Shift Left with Carry — CSL

## Operation




*n* = RS or IMM4

Shifts the bits in register RD *n* positions to the left. The lower *n* bits of the register RD become filled with the carry flag. The carry flag will be updated to the bit contained in RD[16-n] before the shift for *n* > 0. *n* can range from 0 to 16.

In immediate address mode, *n* is determined by the operand IMM4. *n* is considered to be 16 if IMM4 is equal to 0.

In dyadic address mode, *n* is determined by the content of RS. *n* is considered to be 16 if the content of RS is greater than 15.

## CCR Effects

| N | Z | V | C |
|---|---|---|---|
| Δ | Δ | Δ | Δ |

N: Set if bit 15 of the result is set; cleared otherwise.

Z: Set if the result is $0000; cleared otherwise.

V: Set if a two´s complement overflow resulted from the operation; cleared otherwise.
$RD[15]_{old}$ ^ $RD[15]_{new}$

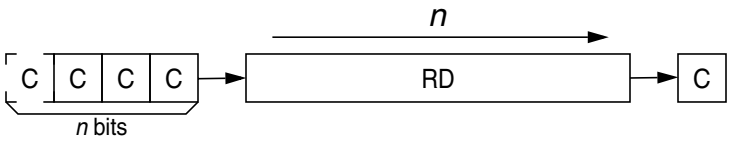
C: Set if *n* > 0 and RD[16-n] = 1; if n = 0 unaffected.

## Code and CPU Cycles

| Source Form | Address Mode | Machine Code | | | | | | | | | | | | Cycles |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| CSL RD, #IMM4 | IMM4 | 0 | 0 | 0 | 0 | 1 | RD | IMM4 | | 1 | 0 | 1 | 0 | P |
| CSL RD, RS | DYA | 0 | 0 | 0 | 0 | 1 | RD | RS | 1 | 0 | 0 | 1 | 0 | P |

# CSR Logical Shift Right with Carry CSR

## Operation




*n* = RS or IMM4

Shifts the bits in register RD *n* positions to the right. The higher *n* bits of the register RD become filled with the carry flag. The carry flag will be updated to the bit contained in RD[n-1] before the shift for *n* > 0. *n* can range from 0 to 16.

In immediate address mode, *n* is determined by the operand IMM4. *n* is considered to be 16 if IMM4 is equal to 0.

In dyadic address mode, *n* is determined by the content of RS. *n* is considered to be 16 if the content of RS is greater than 15.

## CCR Effects

| N | Z | V | C |
|---|---|---|---|
| Δ | Δ | Δ | Δ |

N: Set if bit 15 of the result is set; cleared otherwise.

Z: Set if the result is $0000; cleared otherwise.

V: Set if a two´s complement overflow resulted from the operation; cleared otherwise.
$RD[15]_{old}$ ^ $RD[15]_{new}$

C: Set if *n* > 0 and RD[n-1] = 1; if n = 0 unaffected.

## Code and CPU Cycles

| Source Form | Address Mode | Machine Code | | | | | | | | | | | | Cycles |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| CSR RD, #IMM4 | IMM4 | 0 | 0 | 0 | 0 | 1 | RD | IMM4 | | 1 | 0 | 1 | 1 | P |
| CSR RD, RS | DYA | 0 | 0 | 0 | 0 | 1 | RD | RS | 1 | 0 | 0 | 1 | 1 | P |

# JAL — Jump and Link — JAL

## Operation

PC + $0002 ⇒ RD; RD ⇒ PC

Jumps to the address stored in RD and saves the return address in RD.

## CCR Effects

| N | Z | V | C |
|---|---|---|---|
| — | — | — | — |

N: Not affected.
Z: Not affected.
V: Not affected.
C: Not affected.

## Code and CPU Cycles

| Source Form | Address Mode | Machine Code | | | | | | | | | | | | | | Cycles |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| JAL RD | MON | 0 | 0 | 0 | 0 | 0 | RD | 1 | 1 | 1 | 1 | 0 | 1 | 1 | 0 | PP |

# LDB — Load Byte from Memory (Low Byte) — LDB

## Operation

| | | | | |
|---|---|---|---|---|
| M[RB, #OFFS5] | ⇒ RD.L; | $00 | ⇒ RD.H | |
| M[RB, RI] | ⇒ RD.L; | $00 | ⇒ RD.H | |
| M[RB, RI] | ⇒ RD.L; | $00 | ⇒ RD.H; | RI+1 ⇒ RI;[1] |
| RI-1 | ⇒ RI; | M[RS, RI] | ⇒ RD.L; | $00 ⇒ RD.H |

Loads a byte from memory into the low byte of register RD. The high byte is cleared.

## CCR Effects

| N | Z | V | C |
|---|---|---|---|
| — | — | — | — |

N: Not affected.
Z: Not affected.
V: Not affected.
C: Not affected.

## Code and CPU Cycles

| Source Form | Address Mode | Machine Code | | | | | | | | | | Cycles |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| LDB RD, (RB, #OFFS5) | IDO5 | 0 | 1 | 0 | 0 | 0 | RD | RB | OFFS5 | | | Pr |
| LDB RD, (RS, RI) | IDR | 0 | 1 | 1 | 0 | 0 | RD | RB | RI | 0 | 0 | Pr |
| LDB RD, (RS, RI+) | IDR+ | 0 | 1 | 1 | 0 | 0 | RD | RB | RI | 0 | 1 | Pr |
| LDB RD, (RS, -RI) | -IDR | 0 | 1 | 1 | 0 | 0 | RD | RB | RI | 1 | 0 | Pr |

1. If the same general purpose register is used as index (RI) and destination register (RD), the content of the register will not be incremented after the data move: M[RB, RI] ⇒ RD.L; $00 ⇒ RD.H

# LDH — Load Immediate 8 bit Constant (High Byte) — LDH

## Operation

IMM8 ⇒ RD.H;

Loads an 8 bit immediate constant into the high byte of register RD. The low byte is not affected.

## CCR Effects

| N | Z | V | C |
|---|---|---|---|
| — | — | — | — |

N: Not affected.
Z: Not affected.
V: Not affected.
C: Not affected.

## Code and CPU Cycles

| Source Form | Address Mode | Machine Code | | | | | | | Cycles |
|---|---|---|---|---|---|---|---|---|---|
| LDH RD, #IMM8 | IMM8 | 1 | 1 | 1 | 1 | 1 | RD | IMM8 | P |

# LDL — Load Immediate 8 bit Constant (Low Byte) — LDL

## Operation

IMM8 ⇒ RD.L; $00 ⇒ RD.H

Loads an 8 bit immediate constant into the low byte of register RD. The high byte is cleared.

## CCR Effects

| N | Z | V | C |
|---|---|---|---|
| — | — | — | — |

N: Not affected.
Z: Not affected.
V: Not affected.
C: Not affected.

## Code and CPU Cycles

| Source Form | Address Mode | Machine Code | | | | | | | Cycles |
|---|---|---|---|---|---|---|---|---|---|
| LDL RD, #IMM8 | IMM8 | 1 | 1 | 1 | 1 | 0 | RD | IMM8 | P |

# LDW Load Word from Memory LDW

## Operation

M[RB, #OFFS5] ⇒ RD
M[RB, RI] ⇒ RD
M[RB, RI] ⇒ RD; RI+2 ⇒ RI[1]
RI-2 ⇒ RI; M[RS, RI] ⇒ RD
IMM16 ⇒ RD (translates to LDL RD, #IMM16[7:0]; LDH RD, #IMM16[15:8])

Loads a 16 bit value into the register RD.

## CCR Effects

| N | Z | V | C |
|---|---|---|---|
| — | — | — | — |

N: Not affected.
Z: Not affected.
V: Not affected.
C: Not affected.

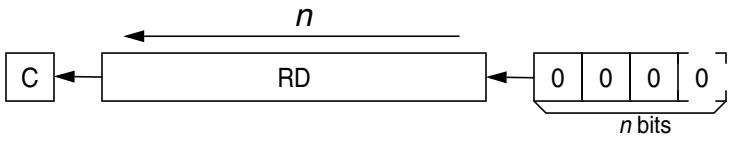
## Code and CPU Cycles

| Source Form | Address Mode | Machine Code | | | | | | | | | | Cycles |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| LDW RD, (RB, #OFFS5) | IDO5 | 0 | 1 | 0 | 0 | 1 | RD | RB | OFFS5 | | | PR |
| LDW RD, (RB, RI) | IDR | 0 | 1 | 1 | 0 | 1 | RD | RB | RI | 0 | 0 | PR |
| LDW RD, (RB, RI+) | IDR+ | 0 | 1 | 1 | 0 | 1 | RD | RB | RI | 0 | 1 | PR |
| LDW RD, (RB, -RI) | -IDR | 0 | 1 | 1 | 0 | 1 | RD | RB | RI | 1 | 0 | PR |
| LDW RD, #IMM16 | IMM8 | 1 | 1 | 1 | 1 | 0 | RD | IMM16[7:0] | | | | P |
| | IMM8 | 1 | 1 | 1 | 1 | 1 | RD | IMM16[15:8] | | | | P |

1. If the same general purpose register is used as index (RI) and destination register (RD), the content of the register will not be incremented after the data move: M[RB, RI] ⇒ RD

# LSL Logical Shift Left LSL

## Operation




*n* = RS or IMM4

Shifts the bits in register RD *n* positions to the left. The lower *n* bits of the register RD become filled with zeros. The carry flag will be updated to the bit contained in RD[16-n] before the shift for *n* > 0.

*n* can range from 0 to 16.

In immediate address mode, *n* is determined by the operand IMM4. *n* is considered to be 16 in IMM4 is equal to 0.

In dyadic address mode, *n* is determined by the content of RS. *n* is considered to be 16 if the content of RS is greater than 15.

## CCR Effects

| N | Z | V | C |
|---|---|---|---|
| Δ | Δ | Δ | Δ |

N: Set if bit 15 of the result is set; cleared otherwise.

Z: Set if the result is $0000; cleared otherwise.

V: Set if a two´s complement overflow resulted from the operation; cleared otherwise.
$RD[15]_{old}$ ^ $RD[15]_{new}$

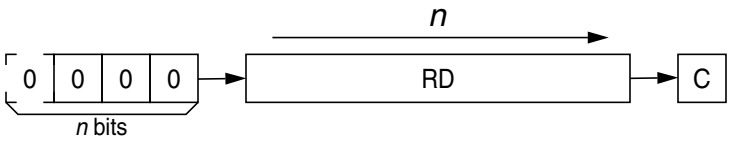
C: Set if *n* > 0 and RD[16-n] = 1; if n = 0 unaffected.

## Code and CPU Cycles

| Source Form | Address Mode | Machine Code | | | | | | | | | | | | Cycles |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| LSL RD, #IMM4 | IMM4 | 0 | 0 | 0 | 0 | 1 | RD | IMM4 | | 1 | 1 | 0 | 0 | P |
| LSL RD, RS | DYA | 0 | 0 | 0 | 0 | 1 | RD | RS | 1 | 0 | 1 | 0 | 0 | P |

# LSR — Logical Shift Right — LSR

## Operation




*n* = RS or IMM4

Shifts the bits in register RD *n* positions to the right. The higher *n* bits of the register RD become filled with zeros. The carry flag will be updated to the bit contained in RD[n-1] before the shift for *n* > 0.

*n* can range from 0 to 16.

In immediate address mode, *n* is determined by the operand IMM4. *n* is considered to be 16 in IMM4 is equal to 0.

In dyadic address mode, *n* is determined by the content of RS. *n* is considered to be 16 if the content of RS is greater than 15.

## CCR Effects

| N | Z | V | C |
|---|---|---|---|
| Δ | Δ | Δ | Δ |

N: Set if bit 15 of the result is set; cleared otherwise.

Z: Set if the result is $0000; cleared otherwise.

V: Set if a two´s complement overflow resulted from the operation; cleared otherwise.
$RD[15]_{old}$ ^ $RD[15]_{new}$

C: Set if *n* > 0 and RD[n-1] = 1; if n = 0 unaffected.

## Code and CPU Cycles

| Source Form | Address Mode | Machine Code | | | | | | | | | | | | Cycles |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| LSR RD, #IMM4 | IMM4 | 0 | 0 | 0 | 0 | 1 | RD | IMM4 | | 1 | 1 | 0 | 1 | P |
| LSR RD, RS | DYA | 0 | 0 | 0 | 0 | 1 | RD | RS | 1 | 0 | 1 | 0 | 1 | P |

# MOV — Move Register Content — MOV

## Operation

RS ⇒ RD (translates to OR RD, R0, RS)

Copies the content of RS to RD.

## CCR Effects

| N | Z | V | C |
|---|---|---|---|
| Δ | Δ | 0 | — |

N: Set if bit 15 of the result is set; cleared otherwise.
Z: Set if the result is $0000; cleared otherwise.
V: 0; cleared.
C: Not affected.

## Code and CPU Cycles

| Source Form | Address Mode | Machine Code | | | | | | | | | | | Cycles |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| MOV RD, RS | TRI | 0 | 0 | 0 | 1 | 0 | RD | 0 | 0 | 0 | RS | 1 | 0 | P |

# NEG — Two's Complement — NEG

## Operation

–RS ⇒ RD (translates to SUB RD, R0, RS)
–RD ⇒ RD (translates to SUB RD, R0, RD)

Performs a two's complement on a general purpose register.

## CCR Effects

| N | Z | V | C |
|---|---|---|---|
| Δ | Δ | Δ | Δ |

N: Set if bit 15 of the result is set; cleared otherwise.
Z: Set if the result is $0000; cleared otherwise.
V: Set if a two´s complement overflow resulted from the operation; cleared otherwise.
$RS[15] \& RD[15]_{new}$
C: Set if there is a carry from the bit 15 of the result; cleared otherwise
$RS[15] \mid RD[15]_{new}$

## Code and CPU Cycles

| Source Form | Address Mode | Machine Code | | | | | | | | | | Cycles |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| NEG RD, RS | TRI | 0 | 0 | 0 | 1 | 1 | RD | 0 | 0 | 0 | RS | 0 | 0 | P |
| NEG RD | TRI | 0 | 0 | 0 | 1 | 1 | RD | 0 | 0 | 0 | RD | 0 | 0 | P |

# NOP — No Operation — NOP

## Operation

No Operation for one cycle.

## CCR Effects

| N | Z | V | C |
|---|---|---|---|
| — | — | — | — |

N: Not affected.
Z: Not affected.
V: Not affected.
C: Not affected.

## Code and CPU Cycles

| Source Form | Address Mode | Machine Code | | | | | | | | | | | | | | | | Cycles |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| NOP | INH | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | P |

# OR — Logical OR — OR

## Operation

RS1 | RS2 ⇒ RD
RD | IMM16⇒ RD (translates to ORL RD, #IMM16[7:0]; ORH RD, #IMM16[15:8]

Performs a bit wise logical OR between two 16 bit values and stores the result in the destination register RD.

**NOTE**

When using immediate addressing mode (OR RD, #IMM16), the Z-flag of the first instruction (ORL RD, #IMM16[7:0]) is not considered by the second instruction (ORH RD, #IMM16[15:8]).
⇒ Don't rely on the Z-Flag.

## CCR Effects

| N | Z | V | C |
|---|---|---|---|
| Δ | Δ | 0 | — |

N: Set if bit 15 of the result is set; cleared otherwise.
Z: Set if the result is $0000; cleared otherwise.
Refer to ORH instruction for #IMM16 operations.
V: 0; cleared.
C: Not affected.

## Code and CPU Cycles

| Source Form | Address Mode | Machine Code | | | | | | | | | | Cycles |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| OR RD, RS1, RS2 | TRI | 0 | 0 | 0 | 1 | 0 | RD | RS1 | RS2 | 1 | 0 | P |
| OR RD, #IMM16 | IMM8 | 1 | 0 | 1 | 0 | 0 | RD | IMM16[7:0] | | | | P |
| | IMM8 | 1 | 0 | 1 | 0 | 1 | RD | IMM16[15:8] | | | | P |

# ORH — Logical OR Immediate 8 bit Constant (High Byte) — ORH

## Operation

RD.H | IMM8 ⇒ RD.H

Performs a bit wise logical OR between the high byte of register RD and an immediate 8 bit constant and stores the result in the destination register RD.H. The low byte of RD is not affected.

## CCR Effects

| N | Z | V | C |
|---|---|---|---|
| Δ | Δ | 0 | — |

N: Set if bit 15 of the result is set; cleared otherwise.
Z: Set if the 8 bit result is $00; cleared otherwise.
V: 0; cleared.
C: Not affected.

## Code and CPU Cycles

| Source Form | Address Mode | Machine Code | | | | | | | Cycles |
|---|---|---|---|---|---|---|---|---|---|
| ORH RD, #IMM8 | IMM8 | 1 | 0 | 1 | 0 | 1 | RD | IMM8 | P |

# ORL — Logical OR Immediate 8 bit Constant (Low Byte) — ORL

## Operation

RD.L | IMM8 ⇒ RD.L

Performs a bit wise logical OR between the low byte of register RD and an immediate 8 bit constant and stores the result in the destination register RD.L. The high byte of RD is not affected.

## CCR Effects

| N | Z | V | C |
|---|---|---|---|
| Δ | Δ | 0 | — |

N: Set if bit 7 of the result is set; cleared otherwise.
Z: Set if the 8 bit result is $00; cleared otherwise.
V: 0; cleared.
C: Not affected.

## Code and CPU Cycles

| Source Form | Address Mode | Machine Code | | | | | | | Cycles |
|---|---|---|---|---|---|---|---|---|---|
| ORL RD, #IMM8 | IMM8 | 1 | 0 | 1 | 0 | 0 | RD | IMM8 | P |

# PAR Calculate Parity PAR

## Operation

Calculates the number of ones in the register RD. The Carry flag will be set if the number is odd, otherwise it will be cleared.

## CCR Effects

| N | Z | V | C |
|---|---|---|---|
| 0 | Δ | 0 | Δ |

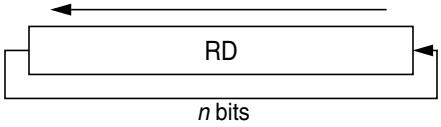
N: 0; cleared.
Z: Set if RD is $0000; cleared otherwise.
V: 0; cleared.
C: Set if the number of ones in the register RD is odd; cleared otherwise.

## Code and CPU Cycles

| Source Form | Address Mode | Machine Code | | | | | | | | | | | | | | Cycles |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| PAR, RD | MON | 0 | 0 | 0 | 0 | 0 | RD | 1 | 1 | 1 | 1 | 0 | 1 | 0 | 1 | P |

# ROL Rotate Left ROL

## Operation




*n* = RS or IMM4

Rotates the bits in register RD *n* positions to the left. The lower *n* bits of the register RD are filled with the upper *n* bits. Two source forms are available. In the first form, the parameter *n* is contained in the instruction code as an immediate operand. In the second form, the parameter is contained in the lower bits of the source register RS[3:0]. All other bits in RS are ignored. If *n* is zero, no shift will take place and the register RD will be unaffected; however, the condition code flags will be updated.

## CCR Effects

| N | Z | V | C |
|---|---|---|---|
| Δ | Δ | 0 | — |

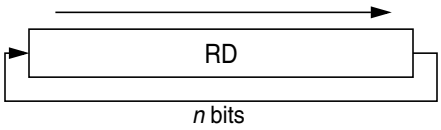
N: Set if bit 15 of the result is set; cleared otherwise.
Z: Set if the result is $0000; cleared otherwise.
V: 0; cleared.
C: Not affected.

## Code and CPU Cycles

| Source Form | Address Mode | Machine Code | | | | | | | | | | | | Cycles |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| ROL RD, #IMM4 | IMM4 | 0 | 0 | 0 | 0 | 1 | RD | IMM4 | | 1 | 1 | 1 | 0 | P |
| ROL RD, RS | DYA | 0 | 0 | 0 | 0 | 1 | RD | RS | 1 | 0 | 1 | 1 | 0 | P |

# ROR — Rotate Right — ROR

## Operation




*n* = RS or IMM4

Rotates the bits in register RD *n* positions to the right. The upper *n* bits of the register RD are filled with the lower *n* bits. Two source forms are available. In the first form, the parameter *n* is contained in the instruction code as an immediate operand. In the second form, the parameter is contained in the lower bits of the source register RS[3:0]. All other bits in RS are ignored. If *n* is zero no shift will take place and the register RD will be unaffected; however, the condition code flags will be updated.

## CCR Effects

| N | Z | V | C |
|---|---|---|---|
| Δ | Δ | 0 | — |

N: Set if bit 15 of the result is set; cleared otherwise.
Z: Set if the result is $0000; cleared otherwise.
V: 0; cleared.
C: Not affected.

## Code and CPU Cycles

| Source Form | Address Mode | Machine Code | | | | | | | | | | | | Cycles |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| ROR RD, #IMM4 | IMM4 | 0 | 0 | 0 | 0 | 1 | RD | IMM4 | | 1 | 1 | 1 | 1 | P |
| ROR RD, RS | DYA | 0 | 0 | 0 | 0 | 1 | RD | RS | 1 | 0 | 1 | 1 | 1 | P |

# RTS — Return to Scheduler — RTS

## Operation

Terminates the current thread of program execution.

## CCR Effects

| N | Z | V | C |
|---|---|---|---|
| — | — | — | — |

N: Not affected.
Z: Not affected.
V: Not affected.
C: Not affected.

## Code and CPU Cycles

| Source Form | Address Mode | Machine Code | | | | | | | | | | | | | | | | Cycles |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| RTS | INH | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | PA |

# SBC — Subtract with Carry — SBC

## Operation

RS1 - RS2 - C ⇒ RD

Subtracts the content of register RS2 and the value of the Carry bit from the content of register RS1 using binary subtraction and stores the result in the destination register RD. Also the zero flag is carried forward from the previous operation allowing 32 and more bit subtractions.

Example:

```
SUB     R6,R4,R2
SBC     R7,R5,R3          ; R7:R6 = R5:R4 - R3:R2
BCC                       ; conditional branch on 32 bit subtraction
```

## CCR Effects

| N | Z | V | C |
|---|---|---|---|
| Δ | Δ | Δ | Δ |

N: Set if bit 15 of the result is set; cleared otherwise.

Z: Set if the result is $0000 and Z was set before this operation; cleared otherwise.

V: Set if a two´s complement overflow resulted from the operation; cleared otherwise.
$RS1[15] \& \overline{RS2[15]} \& \overline{RD[15]}_{new} \mid \overline{RS1[15]} \& RS2[15] \& RD[15]_{new}$

C: Set if there is a carry from bit 15 of the result; cleared otherwise.
$\overline{RS1[15]} \& RS2[15] \mid \overline{RS1[15]} \& RD[15]_{new} \mid RS2[15] \& RD[15]_{new}$

## Code and CPU Cycles

| Source Form | Address Mode | Machine Code | | | | | | | | | | Cycles |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| SBC RD, RS1, RS2 | TRI | 0 | 0 | 0 | 1 | 1 | RD | RS1 | RS2 | 0 | 1 | P |

# SEX — Sign Extend Byte to Word — SEX

## Operation

The result in RD is the 16 bit sign extended representation of the original two's complement number in the low byte of RD.L.

## CCR Effects

| N | Z | V | C |
|---|---|---|---|
| Δ | Δ | 0 | — |

N: Set if bit 15 of the result is set; cleared otherwise.
Z: Set if the result is $0000; cleared otherwise.
V: 0; cleared.
C: Not affected.

## Code and CPU Cycles

| Source Form | Address Mode | Machine Code | | | | | | | | | | | | | | Cycles |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| SEX RD | MON | 0 | 0 | 0 | 0 | 0 | RD | 1 | 1 | 1 | 1 | 0 | 1 | 0 | 0 | P |

# SIF — Set Interrupt Flag — SIF

## Operation

Sets the interrupt flag of an XGATE channel (XGIF). This instruction supports two source forms. If inherent address mode is used, then the interrupt flag of the current channel (XGCHID) will be set. If the monadic address form is used, the interrupt flag associated with the channel id number contained in RS[6:0] is set. The content of RS[15:7] is ignored.

**NOTE**

Interrupt flags of reserved channels (see Device User Guide) can't be set.

## CCR Effects

| N | Z | V | C |
|---|---|---|---|
| — | — | — | — |

N: Not affected.
Z: Not affected.
V: Not affected.
C: Not affected.

## Code and CPU Cycles

| Source Form | Address Mode | Machine Code | | | | | | | | | | | | | | | | Cycles |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| SIF | INH | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | PA |
| SIF RS | MON | 0 | 0 | 0 | 0 | 0 | RS | | | 1 | 1 | 1 | 1 | 0 | 1 | 1 | 1 | PA |

# SSEM Set Semaphore SSEM

## Operation

Attempts to set a semaphore. The state of the semaphore will be stored in the Carry-Flag:

1 = Semaphore is locked by the RISC core

0 = Semaphore is locked by the S12X_CPU

In monadic address mode, bits RS[2:0] select the semaphore to be set.

## CCR Effects

| N | Z | V | C |
|---|---|---|---|
| — | — | — | Δ |

N: Not affected.
Z: Not affected.
V: Not affected.
C: Set if semaphore is locked by the RISC core; cleared otherwise.

## Code and CPU Cycles

| Source Form | Address Mode | Machine Code | | | | | | | | | | | | | | Cycles |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| SSEM #IMM3 | IMM3 | 0 | 0 | 0 | 0 | 0 | IMM3 | 1 | 1 | 1 | 1 | 0 | 0 | 1 | 0 | PA |
| SSEM RS | MON | 0 | 0 | 0 | 0 | 0 | RS | 1 | 1 | 1 | 1 | 0 | 0 | 1 | 1 | PA |

# STB — Store Byte to Memory (Low Byte) — STB

## Operation

RS.L ⇒ M[RB, #OFFS5]
RS.L ⇒ M[RB, RI]
RS.L ⇒ M[RB, RI]; RI+1 ⇒ RI;
RI–1 ⇒ RI; RS.L ⇒ M[RB, RI][1]

Stores the low byte of register RS to memory.

## CCR Effects

| N | Z | V | C |
|---|---|---|---|
| — | — | — | — |

N: Not affected.
Z: Not affected.
V: Not affected.
C: Not affected.

## Code and CPU Cycles

| Source Form | Address Mode | Machine Code | | | | | | | | | | Cycles |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| STB RS, (RB, #OFFS5), | IDO5 | 0 | 1 | 0 | 1 | 0 | RS | RB | OFFS5 | | | Pw |
| STB RS, (RB, RI) | IDR | 0 | 1 | 1 | 1 | 0 | RS | RB | RI | 0 | 0 | Pw |
| STB RS, (RB, RI+) | IDR+ | 0 | 1 | 1 | 1 | 0 | RS | RB | RI | 0 | 1 | Pw |
| STB RS, (RB, -RI) | -IDR | 0 | 1 | 1 | 1 | 0 | RS | RB | RI | 1 | 0 | Pw |

1. If the same general purpose register is used as index (RI) and source register (RS), the unmodified content of the source register is written to the memory: RS.L ⇒ M[RB, RS-1]; RS-1 ⇒ RS

# STW Store Word to Memory STW

## Operation

RS ⇒ M[RB, #OFFS5]
RS ⇒ M[RB, RI]
RS ⇒ M[RB, RI]; RI+2 ⇒ RI;
RI–2 ⇒ RI; RS ⇒ M[RB, RI][1]

Stores the content of register RS to memory.

## CCR Effects

| N | Z | V | C |
|---|---|---|---|
| — | — | — | — |

N: Not affected.
Z: Not affected.
V: Not affected.
C: Not affected.

## Code and CPU Cycles

| Source Form | Address Mode | Machine Code | | | | | | | | | | Cycles |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| STW RS, (RB, #OFFS5) | IDO5 | 0 | 1 | 0 | 1 | 1 | RS | RB | OFFS5 | | | PW |
| STW RS, (RB, RI) | IDR | 0 | 1 | 1 | 1 | 1 | RS | RB | RI | 0 | 0 | PW |
| STW RS, (RB, RI+) | IDR+ | 0 | 1 | 1 | 1 | 1 | RS | RB | RI | 0 | 1 | PW |
| STW RS, (RB, -RI) | -IDR | 0 | 1 | 1 | 1 | 1 | RS | RB | RI | 1 | 0 | PW |

1. If the same general purpose register is used as index (RI) and source register (RS), the unmodified content of the source register is written to the memory: RS ⇒ M[RB, RS–2]; RS–2 ⇒ RS

# SUB — Subtract without Carry — SUB

## Operation

RS1 – RS2 ⇒ RD
RD – IMM16 ⇒ RD (translates to SUBL RD, #IMM16[7:0]; SUBH RD, #IMM16{15:8])

Subtracts two 16 bit values and stores the result in the destination register RD.

**NOTE**

When using immediate addressing mode (SUB RD, #IMM16), the V-flag and the C-Flag of the first instruction (SUBL RD, #IMM16[7:0]) are not considered by the second instruction (SUBH RD, #IMM16[15:8]).
⇒ Don't rely on the V-Flag if RD - IMM16[7:0] < $-2^{15}$.
⇒ Don't rely on the C-Flag if RD < IMM16[7:0].

## CCR Effects

| N | Z | V | C |
|---|---|---|---|
| Δ | Δ | Δ | Δ |

N: Set if bit 15 of the result is set; cleared otherwise.

Z: Set if the result is $0000; cleared otherwise.

V: Set if a two´s complement overflow resulted from the operation; cleared otherwise.
$RS1[15] \& \overline{RS2[15]} \& \overline{RD[15]}_{new} \mid \overline{RS1[15]} \& RS2[15] \& RD[15]_{new}$
Refer to SUBH instruction for #IMM16 operations.

C: Set if there is a carry from the bit 15 of the result; cleared otherwise.
$\overline{RS1[15]} \& RS2[15] \mid \overline{RS1[15]} \& RD[15]_{new} \mid RS2[15] \& RD[15]_{new}$
Refer to SUBH instruction for #IMM16 operations.

## Code and CPU Cycles

| Source Form | Address Mode | Machine Code | | | | | | | | | | Cycles |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| SUB RD, RS1, RS2 | TRI | 0 | 0 | 0 | 1 | 1 | RD | RS1 | RS2 | 0 | 0 | P |
| SUB RD, #IMM16 | IMM8 | 1 | 1 | 0 | 0 | 0 | RD | IMM16[7:0] | | | | P |
| | IMM8 | 1 | 1 | 0 | 0 | 1 | RD | IMM16[15:8] | | | | P |

# SUBH — Subtract Immediate 8 bit Constant (High Byte) — SUBH

## Operation

RD – IMM8:$00 ⇒ RD

Subtracts a signed immediate 8 bit constant from the content of high byte of register RD and using binary subtraction and stores the result in the high byte of destination register RD. This instruction can be used after an SUBL for a 16 bit immediate subtraction.

Example:

```
SUBL    R2,#LOWBYTE
SUBH    R2,#HIGHBYTE      ; R2 = R2 - 16 bit immediate
```

## CCR Effects

| N | Z | V | C |
|---|---|---|---|
| Δ | Δ | Δ | Δ |

N: Set if bit 15 of the result is set; cleared otherwise.

Z: Set if the result is $0000; cleared otherwise.

V: Set if a two´s complement overflow resulted from the operation; cleared otherwise.
$RD[15]_{old} \& \overline{IMM8[7]} \& \overline{RD[15]}_{new} \mid \overline{RD[15]}_{old} \& IMM8[7] \& RD[15]_{new}$

C: Set if there is a carry from the bit 15 of the result; cleared otherwise.
$\overline{RD[15]}_{old} \& IMM8[7] \mid \overline{RD[15]}_{old} \& RD[15]_{new} \mid IMM8[7] \& RD[15]_{new}$

## Code and CPU Cycles

| Source Form | Address Mode | Machine Code | | | | | | | Cycles |
|---|---|---|---|---|---|---|---|---|---|
| SUBH RD, #IMM8 | IMM8 | 1 | 1 | 0 | 0 | 1 | RD | IMM8 | P |

# SUBL — Subtract Immediate 8 bit Constant (Low Byte) — SUBL

## Operation

RD – $00:IMM8 ⇒ RD

Subtracts an immediate 8 bit constant from the content of register RD using binary subtraction and stores the result in the destination register RD.

## CCR Effects

| N | Z | V | C |
|---|---|---|---|
| Δ | Δ | Δ | Δ |

N: Set if bit 15 of the result is set; cleared otherwise.

Z: Set if the result is $0000; cleared otherwise.

V: Set if a two´s complement overflow resulted from the 8 bit operation; cleared otherwise.
$RD[15]_{old}$ & $\overline{RD[15]}_{new}$

C: Set if there is a carry from the bit 15 of the result; cleared otherwise.
$\overline{RD[15]}_{old}$ & $RD[15]_{new}$

## Code and CPU Cycles

| Source Form | Address Mode | Machine Code | | | | | | | Cycles |
|---|---|---|---|---|---|---|---|---|---|
| SUBL RD, #IMM8 | IMM8 | 1 | 1 | 0 | 0 | 0 | RD | IMM8 | P |

# TFR — Transfer from and to Special Registers — TFR

## Operation

TFR RD,CCR: CCR ⇒ RD[3:0]; 0 ⇒ RD[15:4]
TFR CCR,RD: RD[3:0] ⇒ CCR
TFR RD,PC: PC+4 ⇒ RD

Transfers the content of one RISC core register to another.
The TFR RD,PC instruction can be used to implement relative subroutine calls.

Example:

```
        TFR     R7,PC    ;Return address (RETADDR) is stored in R7
        BRA     SUBR     ;Relative branch to subroutine (SUBR)
RETADDR ...

SUBR    ...
        JAL     R7       ;Jump to return address (RETADDR)
```

## CCR Effects

TFR RD,CCR, TFR RD,PC:

| N | Z | V | C |
|---|---|---|---|
| — | — | — | — |

N: Not affected.
Z: Not affected.
V: Not affected.
C: Not affected.

TFR CCR,RS:

| N | Z | V | C |
|---|---|---|---|
| Δ | Δ | Δ | Δ |

N: RS[3].
Z: RS[2].
V: RS[1].
C: RS[0].

## Code and CPU Cycles

| Source Form | Address Mode | Machine Code | | | | | | | | | | | | | | Cycles |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| TFR RD,CCR CCR ⇒ RD | MON | 0 | 0 | 0 | 0 | 0 | RD | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | P |
| TFR CCR,RS RS ⇒ CCR | MON | 0 | 0 | 0 | 0 | 0 | RS | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 1 | P |
| TFR RD,PCPC+4 ⇒ RD | MON | 0 | 0 | 0 | 0 | 0 | RD | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 0 | P |

# TST Test Register TST

## Operation

RS – 0 ⇒ NONE (translates to SUB R0, RS, R0)

Subtracts zero from the content of register RS using binary subtraction and discards the result.

## CCR Effects

| N | Z | V | C |
|---|---|---|---|
| Δ | Δ | Δ | Δ |

N: Set if bit 15 of the result is set; cleared otherwise.

Z: Set if the result is $0000; cleared otherwise.

V: Set if a two´s complement overflow resulted from the operation; cleared otherwise.
RS[15] & $\overline{\text{result[15]}}$

C: Set if there is a carry from the bit 15 of the result; cleared otherwise.
$\overline{\text{RS1[15]}}$ & result[15]

## Code and CPU Cycles

| Source Form | Address Mode | Machine Code | | | | | | | | | | | | Cycles |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| TST RS | TRI | 0 | 0 | 0 | 1 | 1 | 0 0 0 | RS1 | 0 0 0 | | | 0 | 0 | P |

# XNOR Logical Exclusive NOR XNOR

## Operation

~(RS1 ^ RS2) ⇒ RD
~(RD ^ IMM16)⇒ RD
(translates to XNOR RD, #IMM16{15:8]; XNOR RD, #IMM16[7:0])

Performs a bit wise logical exclusive NOR between two 16 bit values and stores the result in the destination register RD.

Remark: Using R0 as a source registers will calculate the one's complement of the other source register. Using R0 as both source operands will fill RD with $FFFF.

**NOTE**

When using immediate addressing mode (XNOR RD, #IMM16), the Z-flag of the first instruction (XNORL RD, #IMM16[7:0]) is not considered by the second instruction (XNORH RD, #IMM16[15:8]).
⇒ Don't rely on the Z-Flag.

## CCR Effects

| N | Z | V | C |
|---|---|---|---|
| Δ | Δ | 0 | — |

N: Set if bit 15 of the result is set; cleared otherwise.
Z: Set if the result is $0000; cleared otherwise.
Refer to XNORH instruction for #IMM16 operations.
V: 0; cleared.
C: Not affected.

## Code and CPU Cycles

| Source Form | Address Mode | Machine Code | | | | | | | | | | Cycles |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| XNOR RD, RS1, RS2 | TRI | 0 | 0 | 0 | 1 | 0 | RD | RS1 | RS2 | 1 | 1 | P |
| XNOR RD, #IMM16 | IMM8 | 1 | 0 | 1 | 1 | 0 | RD | IMM16[7:0] | | | | P |
| | IMM8 | 1 | 0 | 1 | 1 | 1 | RD | IMM16[15:8] | | | | P |

# XNORH Logical Exclusive NOR Immediate 8 bit Constant (High Byte) XNORH

## Operation

~(RD.H ^ IMM8) ⇒ RD.H

Performs a bit wise logical exclusive NOR between the high byte of register RD and an immediate 8 bit constant and stores the result in the destination register RD.H. The low byte of RD is not affected.

## CCR Effects

| N | Z | V | C |
|---|---|---|---|
| Δ | Δ | 0 | — |

N: Set if bit 15 of the result is set; cleared otherwise.
Z: Set if the 8 bit result is $00; cleared otherwise.
V: 0; cleared.
C: Not affected.

## Code and CPU Cycles

| Source Form | Address Mode | Machine Code | | | | | | | Cycles |
|---|---|---|---|---|---|---|---|---|---|
| XNORH RD, #IMM8 | IMM8 | 1 | 0 | 1 | 1 | 1 | RD | IMM8 | P |

# XNORL Logical Exclusive NOR Immediate 8 bit Constant (Low Byte) XNORL

## Operation

~(RD.L ^ IMM8) ⇒ RD.L

Performs a bit wise logical exclusive NOR between the low byte of register RD and an immediate 8 bit constant and stores the result in the destination register RD.L. The high byte of RD is not affected.

## CCR Effects

| N | Z | V | C |
|---|---|---|---|
| Δ | Δ | 0 | — |

N: Set if bit 7 of the result is set; cleared otherwise.
Z: Set if the 8 bit result is $00; cleared otherwise.
V: 0; cleared.
C: Not affected.

## Code and CPU Cycles

| Source Form | Address Mode | Machine Code | | | | | | | Cycles |
|---|---|---|---|---|---|---|---|---|---|
| XNORL RD, #IMM8 | IMM8 | 1 | 0 | 1 | 1 | 0 | RD | IMM8 | P |

## 10.8.6 Instruction Coding

Table 10-24 summarizes all XGATE instructions in the order of their machine coding.

**Table 10-24. Instruction Set Summary (Sheet 1 of 3)**

| Functionality | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **Return to Scheduler and Others** | | | | | | | | | | | | | | | | |
| BRK | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| NOP | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| RTS | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| SIF | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| **Semaphore Instructions** | | | | | | | | | | | | | | | | |
| CSEM IMM3 | 0 | 0 | 0 | 0 | 0 | IMM3 | | | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 |
| CSEM RS | 0 | 0 | 0 | 0 | 0 | RS | | | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 1 |
| SSEM IMM3 | 0 | 0 | 0 | 0 | 0 | IMM3 | | | 1 | 1 | 1 | 1 | 0 | 0 | 1 | 0 |
| SSEM RS | 0 | 0 | 0 | 0 | 0 | RS | | | 1 | 1 | 1 | 1 | 0 | 0 | 1 | 1 |
| **Single Register Instructions** | | | | | | | | | | | | | | | | |
| SEX RD | 0 | 0 | 0 | 0 | 0 | RD | | | 1 | 1 | 1 | 1 | 0 | 1 | 0 | 0 |
| PAR RD | 0 | 0 | 0 | 0 | 0 | RD | | | 1 | 1 | 1 | 1 | 0 | 1 | 0 | 1 |
| JAL RD | 0 | 0 | 0 | 0 | 0 | RD | | | 1 | 1 | 1 | 1 | 0 | 1 | 1 | 0 |
| SIF RS | 0 | 0 | 0 | 0 | 0 | RS | | | 1 | 1 | 1 | 1 | 0 | 1 | 1 | 1 |
| **Special Move instructions** | | | | | | | | | | | | | | | | |
| TFR RD,CCR | 0 | 0 | 0 | 0 | 0 | RD | | | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 |
| TFR CCR,RS | 0 | 0 | 0 | 0 | 0 | RS | | | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 1 |
| TFR RD,PC | 0 | 0 | 0 | 0 | 0 | RD | | | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 0 |
| **Shift instructions Dyadic** | | | | | | | | | | | | | | | | |
| BFFO RD, RS | 0 | 0 | 0 | 0 | 1 | RD | | | RS | | | 1 | 0 | 0 | 0 | 0 |
| ASR RD, RS | 0 | 0 | 0 | 0 | 1 | RD | | | RS | | | 1 | 0 | 0 | 0 | 1 |
| CSL RD, RS | 0 | 0 | 0 | 0 | 1 | RD | | | RS | | | 1 | 0 | 0 | 1 | 0 |
| CSR RD, RS | 0 | 0 | 0 | 0 | 1 | RD | | | RS | | | 1 | 0 | 0 | 1 | 1 |
| LSL RD, RS | 0 | 0 | 0 | 0 | 1 | RD | | | RS | | | 1 | 0 | 1 | 0 | 0 |
| LSR RD, RS | 0 | 0 | 0 | 0 | 1 | RD | | | RS | | | 1 | 0 | 1 | 0 | 1 |
| ROL RD, RS | 0 | 0 | 0 | 0 | 1 | RD | | | RS | | | 1 | 0 | 1 | 1 | 0 |
| ROR RD, RS | 0 | 0 | 0 | 0 | 1 | RD | | | RS | | | 1 | 0 | 1 | 1 | 1 |
| **Shift instructions immediate** | | | | | | | | | | | | | | | | |
| ASR RD, #IMM4 | 0 | 0 | 0 | 0 | 1 | RD | | | IMM4 | | | | 1 | 0 | 0 | 1 |
| CSL RD, #IMM4 | 0 | 0 | 0 | 0 | 1 | RD | | | IMM4 | | | | 1 | 0 | 1 | 0 |
| CSR RD, #IMM4 | 0 | 0 | 0 | 0 | 1 | RD | | | IMM4 | | | | 1 | 0 | 1 | 1 |
| LSL RD, #IMM4 | 0 | 0 | 0 | 0 | 1 | RD | | | IMM4 | | | | 1 | 1 | 0 | 0 |
| LSR RD, #IMM4 | 0 | 0 | 0 | 0 | 1 | RD | | | IMM4 | | | | 1 | 1 | 0 | 1 |
| ROL RD, #IMM4 | 0 | 0 | 0 | 0 | 1 | RD | | | IMM4 | | | | 1 | 1 | 1 | 0 |
| ROR RD, #IMM4 | 0 | 0 | 0 | 0 | 1 | RD | | | IMM4 | | | | 1 | 1 | 1 | 1 |
| **Logical Triadic** | | | | | | | | | | | | | | | | |
| AND RD, RS1, RS2 | 0 | 0 | 0 | 1 | 0 | RD | | | RS1 | | | RS2 | | | 0 | 0 |
| OR RD, RS1, RS2 | 0 | 0 | 0 | 1 | 0 | RD | | | RS1 | | | RS2 | | | 1 | 0 |
| XNOR RD, RS1, RS2 | 0 | 0 | 0 | 1 | 0 | RD | | | RS1 | | | RS2 | | | 1 | 1 |
| **Arithmetic Triadic** | For compare use SUB R0,Rs1,Rs2 | | | | | | | | | | | | | | | |
| SUB RD, RS1, RS2 | 0 | 0 | 0 | 1 | 1 | RD | | | RS1 | | | RS2 | | | 0 | 0 |
| SBC RD, RS1, RS2 | 0 | 0 | 0 | 1 | 1 | RD | | | RS1 | | | RS2 | | | 0 | 1 |
| ADD RD, RS1, RS2 | 0 | 0 | 0 | 1 | 1 | RD | | | RS1 | | | RS2 | | | 1 | 0 |
| ADC RD, RS1, RS2 | 0 | 0 | 0 | 1 | 1 | RD | | | RS1 | | | RS2 | | | 1 | 1 |

**Table 10-24. Instruction Set Summary (Sheet 2 of 3)**

| Functionality | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **Branches** | | | | | | | | | | | | | | | | |
| BCC REL9 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | REL9 | | | | | | | | |
| BCS REL9 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | REL9 | | | | | | | | |
| BNE REL9 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | REL9 | | | | | | | | |
| BEQ REL9 | 0 | 0 | 1 | 0 | 0 | 1 | 1 | REL9 | | | | | | | | |
| BPL REL9 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | REL9 | | | | | | | | |
| BMI REL9 | 0 | 0 | 1 | 0 | 1 | 0 | 1 | REL9 | | | | | | | | |
| BVC REL9 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | REL9 | | | | | | | | |
| BVS REL9 | 0 | 0 | 1 | 0 | 1 | 1 | 1 | REL9 | | | | | | | | |
| BHI REL9 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | REL9 | | | | | | | | |
| BLS REL9 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | REL9 | | | | | | | | |
| BGE REL9 | 0 | 0 | 1 | 1 | 0 | 1 | 0 | REL9 | | | | | | | | |
| BLT REL9 | 0 | 0 | 1 | 1 | 0 | 1 | 1 | REL9 | | | | | | | | |
| BGT REL9 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | REL9 | | | | | | | | |
| BLE REL9 | 0 | 0 | 1 | 1 | 1 | 0 | 1 | REL9 | | | | | | | | |
| BRA REL10 | 0 | 0 | 1 | 1 | 1 | 1 | REL10 | | | | | | | | | |
| **Load and Store Instructions** | | | | | | | | | | | | | | | | |
| LDB RD, (RB, #OFFS5) | 0 | 1 | 0 | 0 | 0 | RD | | | RB | | | OFFS5 | | | | |
| LDW RD, (RB, #OFFS5) | 0 | 1 | 0 | 0 | 1 | RD | | | RB | | | OFFS5 | | | | |
| STB RS, (RB, #OFFS5) | 0 | 1 | 0 | 1 | 0 | RS | | | RB | | | OFFS5 | | | | |
| STW RS, (RB, #OFFS5) | 0 | 1 | 0 | 1 | 1 | RS | | | RB | | | OFFS5 | | | | |
| LDB RD, (RB, RI) | 0 | 1 | 1 | 0 | 0 | RD | | | RB | | | RI | | | 0 | 0 |
| LDW RD, (RB, RI) | 0 | 1 | 1 | 0 | 1 | RD | | | RB | | | RI | | | 0 | 0 |
| STB RS, (RB, RI) | 0 | 1 | 1 | 1 | 0 | RS | | | RB | | | RI | | | 0 | 0 |
| STW RS, (RB, RI) | 0 | 1 | 1 | 1 | 1 | RS | | | RB | | | RI | | | 0 | 0 |
| LDB RD, (RB, RI+) | 0 | 1 | 1 | 0 | 0 | RD | | | RB | | | RI | | | 0 | 1 |
| LDW RD, (RB, RI+) | 0 | 1 | 1 | 0 | 1 | RD | | | RB | | | RI | | | 0 | 1 |
| STB RS, (RB, RI+) | 0 | 1 | 1 | 1 | 0 | RS | | | RB | | | RI | | | 0 | 1 |
| STW RS, (RB, RI+) | 0 | 1 | 1 | 1 | 1 | RS | | | RB | | | RI | | | 0 | 1 |
| LDB RD, (RB, –RI) | 0 | 1 | 1 | 0 | 0 | RD | | | RB | | | RI | | | 1 | 0 |
| LDW RD, (RB, –RI) | 0 | 1 | 1 | 0 | 1 | RD | | | RB | | | RI | | | 1 | 0 |
| STB RS, (RB, –RI) | 0 | 1 | 1 | 1 | 0 | RS | | | RB | | | RI | | | 1 | 0 |
| STW RS, (RB, –RI) | 0 | 1 | 1 | 1 | 1 | RS | | | RB | | | RI | | | 1 | 0 |
| **Bit Field Instructions** | | | | | | | | | | | | | | | | |
| BFEXT RD, RS1, RS2 | 0 | 1 | 1 | 0 | 0 | RD | | | RS1 | | | RS2 | | | 1 | 1 |
| BFINS RD, RS1, RS2 | 0 | 1 | 1 | 0 | 1 | RD | | | RS1 | | | RS2 | | | 1 | 1 |
| BFINSI RD, RS1, RS2 | 0 | 1 | 1 | 1 | 0 | RD | | | RS1 | | | RS2 | | | 1 | 1 |
| BFINSX RD, RS1, RS2 | 0 | 1 | 1 | 1 | 1 | RD | | | RS1 | | | RS2 | | | 1 | 1 |
| **Logic Immediate Instructions** | | | | | | | | | | | | | | | | |
| ANDL RD, #IMM8 | 1 | 0 | 0 | 0 | 0 | RD | | | IMM8 | | | | | | | |
| ANDH RD, #IMM8 | 1 | 0 | 0 | 0 | 1 | RD | | | IMM8 | | | | | | | |
| BITL RD, #IMM8 | 1 | 0 | 0 | 1 | 0 | RD | | | IMM8 | | | | | | | |
| BITH RD, #IMM8 | 1 | 0 | 0 | 1 | 1 | RD | | | IMM8 | | | | | | | |
| ORL RD, #IMM8 | 1 | 0 | 1 | 0 | 0 | RD | | | IMM8 | | | | | | | |
| ORH RD, #IMM8 | 1 | 0 | 1 | 0 | 1 | RD | | | IMM8 | | | | | | | |
| XNORL RD, #IMM8 | 1 | 0 | 1 | 1 | 0 | RD | | | IMM8 | | | | | | | |
| XNORH RD, #IMM8 | 1 | 0 | 1 | 1 | 1 | RD | | | IMM8 | | | | | | | |

**Table 10-24. Instruction Set Summary (Sheet 3 of 3)**

| Functionality | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **Arithmetic Immediate Instructions** | | | | | | | | | | | | | | | | |
| SUBL RD, #IMM8 | 1 | 1 | 0 | 0 | 0 | RD | | | IMM8 | | | | | | | |
| SUBH RD, #IMM8 | 1 | 1 | 0 | 0 | 1 | RD | | | IMM8 | | | | | | | |
| CMPL RS, #IMM8 | 1 | 1 | 0 | 1 | 0 | RS | | | IMM8 | | | | | | | |
| CPCH RS, #IMM8 | 1 | 1 | 0 | 1 | 1 | RS | | | IMM8 | | | | | | | |
| ADDL RD, #IMM8 | 1 | 1 | 1 | 0 | 0 | RD | | | IMM8 | | | | | | | |
| ADDH RD, #IMM8 | 1 | 1 | 1 | 0 | 1 | RD | | | IMM8 | | | | | | | |
| LDL RD, #IMM8 | 1 | 1 | 1 | 1 | 0 | RD | | | IMM8 | | | | | | | |
| LDH RD, #IMM8 | 1 | 1 | 1 | 1 | 1 | RD | | | IMM8 | | | | | | | |

## 10.9 Initialization and Application Information

### 10.9.1 Initialization

The recommended initialization of the XGATE is as follows:

1. Clear the XGE bit to suppress any incoming service requests.
2. Make sure that no thread is running on the XGATE. This can be done in several ways:
   a) Poll the XGCHID register until it reads $00. Also poll XGDBG and XGSWEF to make sure that the XGATE has not been stopped.
   b) Enter Debug Mode by setting the XGDBG bit. Clear the XGCHID register. Clear the XGDBG bit.

   The recommended method is a).
3. Set the XGVBR register to the lowest address of the XGATE vector space.
4. Clear all Channel ID flags.
5. Copy XGATE vectors and code into the RAM.
6. Initialize the S12X_INT module.
7. Enable the XGATE by setting the XGE bit.

The following code example implements the XGATE initialization sequence.

### 10.9.2 Code Example (Transmit "Hello World!" on SCI)

```
                        CPU   S12X
                        ;###########################################
                        ;#                  SYMBOLS                #
                        ;###########################################
SCI_REGS                EQU   $00C8           ;SCI register space
SCIBDH                  EQU   SCI_REGS+$00;   ;SCI Baud Rate Register
SCIBDL                  EQU   SCI_REGS+$00    ;SCI Baud Rate Register
SCICR2                  EQU   SCI_REGS+$03    ;SCI Control Register 2
SCISR1                  EQU   SCI_REGS+$04    ;SCI Status  Register 1
SCIDRL                  EQU   SCI_REGS+$07    ;SCI Control Register 2
TIE                     EQU   $80             ;TIE bit mask
TE                      EQU   $08             ;TE bit mask
RE                      EQU   $04             ;RE bit mask
```

```
SCI_VEC                 EQU  $D6             ;SCI vector number

INT_REGS                EQU  $0120           ;S12X_INT register space
INT_CFADDR              EQU  INT_REGS+$07    ;Interrupt Configuration Address Register
INT_CFDATA              EQU  INT_REGS+$08    ;Interrupt Configuration Data Registers
RQST                    EQU  $80             ;RQST bit mask

XGATE_REGS              EQU  $0380           ;XGATE register space
XGMCTL                  EQU  XGATE_REGS+$00  ;XGATE Module Control Register
XGMCTL_CLEAR            EQU  $FA02           ;Clear all XGMCTL bits
XGMCTL_ENABLE           EQU  $8282           ;Enable XGATE
XGCHID                  EQU  XGATE_REGS+$02  ;XGATE Channel ID Register
XGISPSEL                EQU  XGATE_REGS+$05  ;XGATE Channel ID Register
XGVBR                   EQU  XGATE_REGS+$06  ;XGATE ISP Select Register
XGIF                    EQU  XGATE_REGS+$08  ;XGATE Interrupt Flag Vector
XGSWT                   EQU  XGATE_REGS+$18  ;XGATE Software Trigger Register
XGSEM                   EQU  XGATE_REGS+$1A  ;XGATE Semaphore Register

RPAGE                   EQU  $0016

RAM_SIZE                EQU  32*$400         ;32k RAM

RAM_START               EQU  $1000
RAM_START_XG            EQU  $10000-RAM_SIZE
RAM_START_GLOB          EQU  $100000-RAM_SIZE

XGATE_VECTORS           EQU  RAM_START
XGATE_VECTORS_XG        EQU  RAM_START_XG

XGATE_DATA              EQU  RAM_START+(4*128)
XGATE_DATA_XG           EQU  RAM_START_XG+(4*128)

XGATE_CODE              EQU  XGATE_DATA+(XGATE_CODE_FLASH-XGATE_DATA_FLASH)
XGATE_CODE_XG           EQU  XGATE_DATA_XG+(XGATE_CODE_FLASH-XGATE_DATA_FLASH)

BUS_FREQ_HZ             EQU  40000000

                        ;###########################################
                        ;#           S12XE VECTOR TABLE            #
                        ;###########################################
                        ORG  $FF10           ;non-maskable interrupts
                        DW   DUMMY_ISR DUMMY_ISR DUMMY_ISR DUMMY_ISR

                        ORG  $FFF4           ;non-maskable interrupts
                        DW   DUMMY_ISR DUMMY_ISR DUMMY_ISR

                        ORG  $FFFA           ;resets
                        DW   START_OF_CODE START_OF_CODE START_OF_CODE

                        ;###########################################
                        ;#                DISABLE COP              #
                        ;###########################################
                        ORG  $FF0E
                        DW   $FFFE

                        ORG  $C000
START_OF_CODE
```

```
                    ;##########################################
                    ;#          INITIALIZE S12XE CORE         #
                    ;##########################################
                    SEI
                    MOVB  #(RAM_START_GLOB>>12), RPAGE    ;set RAM page

                    ;##########################################
                    ;#              INITIALIZE SCI            #
                    ;##########################################
INIT_SCI            MOVW  #(BUS_FREQ_HZ/(16*9600)), SCIBDH;set baud rate
                    MOVB  #(TIE|TE), SCICR2;enable tx buffer empty interrupt

                    ;##########################################
                    ;#           INITIALIZE S12X_INT          #
                    ;##########################################
INIT_INT            MOVB  #(SCI_VEC&$F0), INT_CFADDR      ;switch SCI interrupts to XGATE
                    MOVB  #RQST|$01, INT_CFDATA+((SCI_VEC&$0F)>>1)

                    ;##########################################
                    ;#             INITIALIZE XGATE           #
                    ;##########################################
INIT_XGATE          MOVW  #XGMCTL_CLEAR, XGMCTL           ;clear all XGMCTL bits

INIT_XGATE_BUSY_LOOP    TST   XGCHID          ;wait until current thread is finished
                    BNE   INIT_XGATE_BUSY_LOOP

                    LDX   #XGIF           ;clear all channel interrupt flags
                    LDD   #$FFFF
                    STD   2,X+
                    STD   2,X+
                    STD   2,X+
                    STD   2,X+
                    STD   2,X+
                    STD   2,X+
                    STD   2,X+
                    STD   2,X+

                    CLR   XGISPSEL        ;set vector base register
                    MOVW  #XGATE_VECTORS_XG, XGVBR
                    MOVW  #$FF00, XGSWT   ;clear all software triggers

                    ;##########################################
                    ;#      INITIALIZE XGATE VECTOR TABLE     #
                    ;##########################################
                    LDAA  #128            ;build XGATE vector table
                    LDY   #XGATE_VECTORS
INIT_XGATE_VECTAB_LOOP  MOVW  #XGATE_DUMMY_ISR_XG, 4,Y+
                    DBNE  A, INIT_XGATE_VECTAB_LOOP

                    MOVW  #XGATE_CODE_XG, RAM_START+(2*SCI_VEC)
                    MOVW  #XGATE_DATA_XG, RAM_START+(2*SCI_VEC)+2

                    ;##########################################
                    ;#              COPY XGATE CODE           #
                    ;##########################################
COPY_XGATE_CODE         LDX   #XGATE_DATA_FLASH
COPY_XGATE_CODE_LOOP    MOVW  2,X+, 2,Y+
```

```
                        MOVW  2,X+, 2,Y+
                        MOVW  2,X+, 2,Y+
                        MOVW  2,X+, 2,Y+
                        CPX   #XGATE_CODE_FLASH_END
                        BLS   COPY_XGATE_CODE_LOOP

                        ;#########################################
                        ;#               START XGATE             #
                        ;#########################################
START_XGATE             MOVW  #XGMCTL_ENABLE, XGMCTL          ;enable XGATE
                        BRA   *

                        ;#########################################
                        ;#     DUMMY INTERRUPT SERVICE ROUTINE     #
                        ;#########################################
DUMMY_ISR               RTI

                        CPU   XGATE
                        ;#########################################
                        ;#                XGATE DATA               #
                        ;#########################################
                        ALIGN 1
XGATE_DATA_FLASH        EQU   *
XGATE_DATA_SCI          EQU   *-XGATE_DATA_FLASH
                        DW    SCI_REGS                        ;pointer to SCI register space
XGATE_DATA_IDX          EQU   *-XGATE_DATA_FLASH
                        DB    XGATE_DATA_MSG                  ;string pointer
XGATE_DATA_MSG          EQU   *-XGATE_DATA_FLASH
                        FCC   "Hello World!                   ;ASCII string
                        DB    $0D                             ;CR

                        ;#########################################
                        ;#               XGATE CODE              #
                        ;#########################################
                        ALIGN 1
XGATE_CODE_FLASH        LDW   R2,(R1,#XGATE_DATA_SCI)         ;SCI -> R2
                        LDB   R3,(R1,#XGATE_DATA_IDX)         ;msg -> R3
                        LDB   R4,(R1,R3+)                     ;curr. char -> R4
                        STB   R3,(R1,#XGATE_DATA_IDX)         ;R3 -> idx
                        LDB   R0,(R2,#(SCISR1-SCI_REGS))      ;initiate SCI transmit
                        STB   R4,(R2,#(SCIDRL-SCI_REGS))      ;initiate SCI transmit
                        CMPL  R4,#$0D
                        BEQ   XGATE_CODE_DONE
                        RTS
XGATE_CODE_DONE         LDL   R4,#$00                         ;disable SCI interrupts
                        STB   R4,(R2,#(SCICR2-SCI_REGS))
                        LDL   R3,#XGATE_DATA_MSG;reset R3
                        STB   R3,(R1,#XGATE_DATA_IDX)
XGATE_CODE_FLASH_END    RTS
XGATE_DUMMY_ISR_XG      EQU   (XGATE_CODE_FLASH_END-XGATE_CODE_FLASH)+XGATE_CODE_XG
```

### 10.9.3 Stack Support

To simplify the implementation of a program stack the XGATE can be configured to set RISC core register R7 to the beginning of a stack region before executing a thread. Two separate stack regions can be defined: One for threads of priority level 7 to 4 (refer to Section 10.3.1.5, “XGATE Initial Stack Pointer for

Interrupt Priorities 7 to 4 (XGISP74)") and one for threads of priority level 3 to 1 (refer to Section 10.3.1.6, "XGATE Initial Stack Pointer for Interrupt Priorities 3 to 1 (XGISP31)").

# Chapter 11
# S12XE Clocks and Reset Generator (S12XECRGV1)

**Table 11-1. Revision History**

| Revision Number | Revision Date | Sections Affected | Description of Changes |
|---|---|---|---|
| V01.03 | 1 Sep. 2008 | Table 11-14 | added 100MHz example for PLL |
| V01.04 | 20 Nov. 2008 | 11.3.2.4/11-475 | S12XECRG Flags Register: corrected address to Module Base + 0x0003 |
| V01.05 | 19. Sep 2009 | 11.5.1/11-495 | Modified Note below Table 11-17./11-495 |
| V01.06 | 18. Sep 2012 | Table 11-14<br>11.5.1 | Added footnote concerning maximum clock frequencies to table<br>Removed redundant examples from table<br>Replaced reference to MMC documentation |

## 11.1 Introduction

This specification describes the function of the Clocks and Reset Generator (S12XECRG).

### 11.1.1 Features

The main features of this block are:

- Phase Locked Loop (IPLL) frequency multiplier with internal filter
  - Reference divider
  - Post divider
  - Configurable internal filter (no external pin)
  - Optional frequency modulation for defined jitter and reduced emission
  - Automatic frequency lock detector
  - Interrupt request on entry or exit from locked condition
  - Self Clock Mode in absence of reference clock
- System Clock Generator
  - Clock Quality Check
  - User selectable fast wake-up from Stop in Self-Clock Mode for power saving and immediate program execution
  - Clock switch for either Oscillator or PLL based system clocks
- Computer Operating Properly (COP) watchdog timer with time-out clear window.
- System Reset generation from the following possible sources:
  - Power on reset
  - Low voltage reset

  - Illegal address reset
  - COP reset
  - Loss of clock reset
  - External pin reset
- Real-Time Interrupt (RTI)

## 11.1.2 Modes of Operation

This subsection lists and briefly describes all operating modes supported by the S12XECRG.

- Run Mode

  All functional parts of the S12XECRG are running during normal Run Mode. If RTI or COP functionality is required the individual bits of the associated rate select registers (COPCTL, RTICTL) have to be set to a non zero value.
- Wait Mode

  In this mode the IPLL can be disabled automatically depending on the PLLWAI bit.
- Stop Mode

  Depending on the setting of the PSTP bit Stop Mode can be differentiated between Full Stop Mode (PSTP = 0) and Pseudo Stop Mode (PSTP = 1).
  - Full Stop Mode

    The oscillator is disabled and thus all system and core clocks are stopped. The COP and the RTI remain frozen.
  - Pseudo Stop Mode

    The oscillator continues to run and most of the system and core clocks are stopped. If the respective enable bits are set the COP and RTI will continue to run, else they remain frozen.
- Self Clock Mode

  Self Clock Mode will be entered if the Clock Monitor Enable Bit (CME) and the Self Clock Mode Enable Bit (SCME) are both asserted and the clock monitor in the oscillator block detects a loss of clock. As soon as Self Clock Mode is entered the S12XECRG starts to perform a clock quality check. Self Clock Mode remains active until the clock quality check indicates that the required quality of the incoming clock signal is met (frequency and amplitude). Self Clock Mode should be used for safety purposes only. It provides reduced functionality to the MCU in case a loss of clock is causing severe system conditions.

## 11.1.3 Block Diagram

Figure 11-1 shows a block diagram of the S12XECRG.

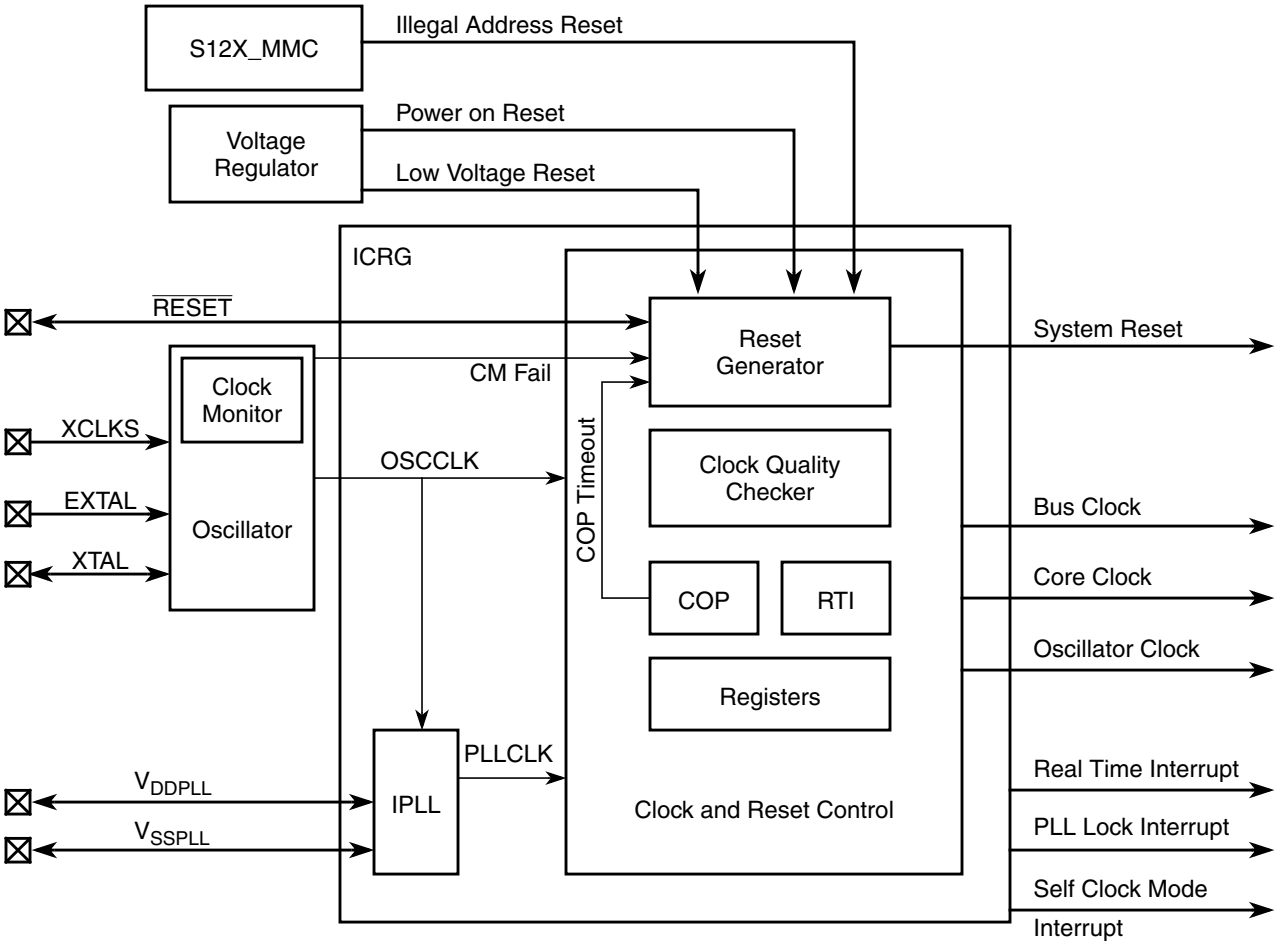



**Figure 11-1. Block diagram of S12XECRG**

## 11.2 Signal Description

This section lists and describes the signals that connect off chip.

### 11.2.1 $V_{DDPLL}$, $V_{SSPLL}$

These pins provides operating voltage ($V_{DDPLL}$) and ground ($V_{SSPLL}$) for the IPLL circuitry. This allows the supply voltage to the IPLL to be independently bypassed. Even if IPLL usage is not required $V_{DDPLL}$ and $V_{SSPLL}$ must be connected to properly.

### 11.2.2 $\overline{\text{RESET}}$

$\overline{\text{RESET}}$ is an active low bidirectional reset pin. As an input it initializes the MCU asynchronously to a known start-up state. As an open-drain output it indicates that an system reset (internal to MCU) has been triggered.

## 11.3 Memory Map and Registers

This section provides a detailed description of all registers accessible in the S12XECRG.

### 11.3.1 Module Memory Map

Figure 11-2 gives an overview on all S12XECRG registers.

| Address | Name | | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|---|---|
| 0x0000 | SYNR | R | VCOFRQ[1:0] | | SYNDIV[5:0] | | | | | |
| | | W | | | | | | | | |
| 0x0001 | REFDV | R | REFFRQ[1:0] | | REFDIV[5:0] | | | | | |
| | | W | | | | | | | | |
| 0x0002 | POSTDIV | R | 0 | 0 | 0 | POSTDIV[4:0] | | | | |
| | | W | | | | | | | | |
| 0x0003 | CRGFLG | R | RTIF | PORF | LVRF | LOCKIF | LOCK | ILAF | SCMIF | SCM |
| | | W | | | | | | | | |
| 0x0004 | CRGINT | R | RTIE | 0 | 0 | LOCKIE | 0 | 0 | SCMIE | 0 |
| | | W | | | | | | | | |
| 0x0005 | CLKSEL | R | PLLSEL | PSTP | XCLKS | 0 | PLLWAI | 0 | RTIWAI | COPWAI |
| | | W | | | | | | | | |
| 0x0006 | PLLCTL | R | CME | PLLON | FM1 | FM0 | FSTWKP | PRE | PCE | SCME |
| | | W | | | | | | | | |
| 0x0007 | RTICTL | R | RTDEC | RTR6 | RTR5 | RTR4 | RTR3 | RTR2 | RTR1 | RTR0 |
| | | W | | | | | | | | |
| 0x0008 | COPCTL | R | WCOP | RSBCK | 0 | 0 | 0 | CR2 | CR1 | CR0 |
| | | W | | | WRTMASK | | | | | |
| 0x0009 | FORBYP[2] | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | W | | | | | | | | |
| 0x000A | CTCTL[2] | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | W | | | | | | | | |
| 0x000B | ARMCOP | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | W | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |

2. FORBYP and CTCTL are intended for factory test purposes only.

= Unimplemented or Reserved

**Figure 11-2. CRG Register Summary**

**NOTE**

Register Address = Base Address + Address Offset, where the Base Address is defined at the MCU level and the Address Offset is defined at the module level.

## 11.3.2 Register Descriptions

This section describes in address order all the S12XECRG registers and their individual bits.

### 11.3.2.1 S12XECRG Synthesizer Register (SYNR)

The SYNR register controls the multiplication factor of the IPLL and selects the VCO frequency range.

Module Base + 0x0000

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | VCOFRQ[1:0] | | SYNDIV[5:0] | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 11-3. S12XECRG Synthesizer Register (SYNR)**

Read: Anytime

Write: Anytime except if PLLSEL = 1

**NOTE**

Write to this register initializes the lock detector bit.

$$f_{VCO} = 2 \times f_{OSC} \times \frac{(SYNDIV + 1)}{(REFDIV + 1)}$$

$$f_{PLL} = \frac{f_{VCO}}{2 \times POSTDIV}$$

$$f_{BUS} = \frac{f_{PLL}}{2}$$

**NOTE**

$f_{VCO}$ must be within the specified VCO frequency lock range. $F._{BUS}$ (Bus Clock) must not exceed the specified maximum. If POSTDIV = $00 then $f_{PLL}$ is same as $f_{VCO}$ (divide by one).

The VCOFRQ[1:0] bit are used to configure the VCO gain for optimal stability and lock time. For correct IPLL operation the VCOFRQ[1:0] bits have to be selected according to the actual target VCOCLK frequency as shown in Table 11-2. Setting the VCOFRQ[1:0] bits wrong can result in a non functional IPLL (no locking and/or insufficient stability).

**Table 11-2. VCO Clock Frequency Selection**

| VCOCLK Frequency Ranges | VCOFRQ[1:0] |
|---|---|
| 32MHz <= $f_{VCO}$<= 48MHz | 00 |
| 48MHz < $f_{VCO}$<= 80MHz | 01 |
| Reserved | 10 |
| 80MHz < $f_{VCO}$ <= 120MHz | 11 |

### 11.3.2.2 S12XECRG Reference Divider Register (REFDV)

The REFDV register provides a finer granularity for the IPLL multiplier steps.

Module Base + 0x0001

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | REFFRQ[1:0] | | REFDIV[5:0] | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 11-4. S12XECRG Reference Divider Register (REFDV)**

Read: Anytime

Write: Anytime except when PLLSEL = 1

**NOTE**

Write to this register initializes the lock detector bit.

$$f_{REF} = \frac{f_{OSC}}{(REFDIV + 1)}$$

The REFFRQ[1:0] bit are used to configure the internal PLL filter for optimal stability and lock time. For correct IPLL operation the REFFRQ[1:0] bits have to be selected according to the actual REFCLK frequency as shown in Figure 11-3. Setting the REFFRQ[1:0] bits wrong can result in a non functional IPLL (no locking and/or insufficient stability).

**Table 11-3. Reference Clock Frequency Selection**

| REFCLK Frequency Ranges | REFFRQ[1:0] |
|---|---|
| 1MHz <= $f_{REF}$ <= 2MHz | 00 |
| 2MHz < $f_{REF}$ <= 6MHz | 01 |
| 6MHz < $f_{REF}$ <= 12MHz | 10 |
| $f_{REF}$ >12MHz | 11 |

### 11.3.2.3 S12XECRG Post Divider Register (POSTDIV)

The POSTDIV register controls the frequency ratio between the VCOCLK and PLLCLK. The count in the final divider divides VCOCLK frequency by 1 or 2*POSTDIV. Note that if POSTDIV = $00 $f_{PLL}$= $f_{VCO}$ (divide by one).

Module Base + 0x0002

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | 0 | POSTDIV[4:0] | | | | |
| W | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented or Reserved

**Figure 11-5. S12XECRG Post Divider Register (POSTDIV)**

Read: Anytime

Write: Anytime except if PLLSEL = 1

$$f_{PLL} = \frac{f_{VCO}}{(2 \times POSTDIV)}$$

**NOTE**

If POSTDIV = $00 then $f_{PLL}$ is identical to $f_{VCO}$ (divide by one).

### 11.3.2.4 S12XECRG Flags Register (CRGFLG)

This register provides S12XECRG status bits and flags.

Module Base + 0x0003

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | RTIF | PORF | LVRF | LOCKIF | LOCK | ILAF | SCMIF | SCM |
| W | | | | | | | | |
| Reset | 0 | Note 1 | Note 2 | Note 3 | 0 | 0 | 0 | 0 |

1. PORF is set to 1 when a power on reset occurs. Unaffected by system reset.
2. LVRF is set to 1 when a low voltage reset occurs. Unaffected by system reset.
3. ILAF is set to 1 when an illegal address reset occurs. Unaffected by system reset. Cleared by power on or low voltage reset.

= Unimplemented or Reserved

**Figure 11-6. S12XECRG Flags Register (CRGFLG)**

Read: Anytime

Write: Refer to each bit for individual write conditions

**Table 11-4. CRGFLG Field Descriptions**

| Field | Description |
|---|---|
| 7<br>RTIF | **Real Time Interrupt Flag** — RTIF is set to 1 at the end of the RTI period. This flag can only be cleared by writing a 1. Writing a 0 has no effect. If enabled (RTIE=1), RTIF causes an interrupt request.<br>0 RTI time-out has not yet occurred.<br>1 RTI time-out has occurred. |
| 6<br>PORF | **Power on Reset Flag** — PORF is set to 1 when a power on reset occurs. This flag can only be cleared by writing a 1. Writing a 0 has no effect.<br>0 Power on reset has not occurred.<br>1 Power on reset has occurred. |
| 5<br>LVRF | **Low Voltage Reset Flag** — LVRF is set to 1 when a low voltage reset occurs. This flag can only be cleared by writing a 1. Writing a 0 has no effect.<br>0 Low voltage reset has not occurred.<br>1 Low voltage reset has occurred. |
| 4<br>LOCKIF | IPLL **Lock Interrupt Flag** — LOCKIF is set to 1 when LOCK status bit changes. This flag can only be cleared by writing a 1. Writing a 0 has no effect.If enabled (LOCKIE=1), LOCKIF causes an interrupt request.<br>0 No change in LOCK bit.<br>1 LOCK bit has changed. |
| 3<br>LOCK | **Lock Status Bit** — LOCK reflects the current state of IPLL lock condition. This bit is cleared in Self Clock Mode. Writes have no effect.<br>0 VCOCLK is not within the desired tolerance of the target frequency.<br>1 VCOCLK is within the desired tolerance of the target frequency. |
| 2<br>ILAF | **Illegal Address Reset Flag** — ILAF is set to 1 when an illegal address reset occurs. Refer to S12XMMC Block Guide for details. This flag can only be cleared by writing a 1. Writing a 0 has no effect.<br>0 Illegal address reset has not occurred.<br>1 Illegal address reset has occurred. |
| 1<br>SCMIF | **Self Clock Mode Interrupt Flag** — SCMIF is set to 1 when SCM status bit changes. This flag can only be cleared by writing a 1. Writing a 0 has no effect. If enabled (SCMIE=1), SCMIF causes an interrupt request.<br>0 No change in SCM bit.<br>1 SCM bit has changed. |
| 0<br>SCM | **Self Clock Mode Status Bit** — SCM reflects the current clocking mode. Writes have no effect.<br>0 MCU is operating normally with OSCCLK available.<br>1 MCU is operating in Self Clock Mode with OSCCLK in an unknown state. All clocks are derived from PLLCLK running at its minimum frequency $f_{SCM}$. |

## 11.3.2.5 S12XECRG Interrupt Enable Register (CRGINT)

This register enables S12XECRG interrupt requests.

Module Base + 0x0004

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | RTIE | 0 | 0 | LOCKIE | 0 | 0 | SCMIE | 0 |
| W | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented or Reserved

**Figure 11-7. S12XECRG Interrupt Enable Register (CRGINT)**

Read: Anytime

Write: Anytime

**Table 11-5. CRGINT Field Descriptions**

| Field | Description |
|---|---|
| 7<br>RTIE | **Real Time Interrupt Enable Bit**<br>0 Interrupt requests from RTI are disabled.<br>1 Interrupt will be requested whenever RTIF is set. |
| 4<br>LOCKIE | **Lock Interrupt Enable Bit**<br>0 LOCK interrupt requests are disabled.<br>1 Interrupt will be requested whenever LOCKIF is set. |
| 1<br>SCMIE | **Self Clock Mode Interrupt Enable Bit**<br>0 SCM interrupt requests are disabled.<br>1 Interrupt will be requested whenever SCMIF is set. |

### 11.3.2.6 S12XECRG Clock Select Register (CLKSEL)

This register controls S12XECRG clock selection. Refer toFigure 11-16 for more details on the effect of each bit.

Module Base + 0x0005

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | PLLSEL | PSTP | XCLKS | 0 | PLLWAI | 0 | RTIWAI | COPWAI |
| W | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented or Reserved

**Figure 11-8. S12XECRG Clock Select Register (CLKSEL)**

Read: Anytime

Write: Refer to each bit for individual write conditions

**Table 11-6. CLKSEL Field Descriptions**

| Field | Description |
|---|---|
| 7<br>PLLSEL | **PLL Select Bit**<br>Write: Anytime.<br>Writing a one when LOCK=0 has no effect. This prevents the selection of an unstable PLLCLK as SYSCLK.<br>PLLSEL bit is cleared when the MCU enters Self Clock Mode, Stop Mode or Wait Mode with PLLWAI bit set.<br>**It is recommended to read back the PLLSEL bit to make sure PLLCLK has really been selected as SYSCLK, as LOCK status bit could theoretically change at the very moment writing the PLLSEL bit.**<br>0 System clocks are derived from OSCCLK ($f_{BUS} = f_{OSC} / 2$).<br>1 System clocks are derived from PLLCLK ($f_{BUS} = f_{PLL} / 2$). |
| 6<br>PSTP | **Pseudo Stop Bit**<br>Write: Anytime<br>This bit controls the functionality of the oscillator during Stop Mode.<br>0 Oscillator is disabled in Stop Mode.<br>1 Oscillator continues to run in Stop Mode (Pseudo Stop).<br>**Note:** Pseudo Stop Mode allows for faster STOP recovery and reduces the mechanical stress and aging of the resonator in case of frequent STOP conditions at the expense of a slightly increased power consumption. |
| 5<br>XCLKS | **Oscillator Configuration Status Bit** — This read-only bit shows the oscillator configuration status.<br>0 Loop controlled Pierce Oscillator is selected.<br>1 External clock / full swing Pierce Oscillator is selected. |
| 3<br>PLLWAI | **PLL Stops in Wait Mode Bit**<br>Write: Anytime<br>If PLLWAI is set, the S12XECRG will clear the PLLSEL bit before entering Wait Mode. The PLLON bit remains set during Wait Mode but the IPLL is powered down. Upon exiting Wait Mode, the PLLSEL bit has to be set manually if PLL clock is required.<br>0 IPLL keeps running in Wait Mode.<br>1 IPLL stops in Wait Mode. |
| 1<br>RTIWAI | **RTI Stops in Wait Mode Bit**<br>Write: Anytime<br>0 RTI keeps running in Wait Mode.<br>1 RTI stops and initializes the RTI dividers whenever the part goes into Wait Mode. |
| 0<br>COPWAI | **COP Stops in Wait Mode Bit**<br>Normal modes: Write once<br>Special modes: Write anytime<br>0 COP keeps running in Wait Mode.<br>1 COP stops and initializes the COP counter whenever the part goes into Wait Mode. |

## 11.3.2.7 S12XECRG IPLL Control Register (PLLCTL)

This register controls the IPLL functionality.

Module Base + 0x0006

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | CME | PLLON | FM1 | FM0 | FSTWKP | PRE | PCE | SCME |
| Reset | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 1 |

**Figure 11-9. S12XECRG IPLL Control Register (PLLCTL)**

Read: Anytime

Write: Refer to each bit for individual write conditions

**Table 11-7. PLLCTL Field Descriptions**

| Field | Description |
|---|---|
| 7<br>CME | **Clock Monitor Enable Bit** — CME enables the clock monitor. Write anytime except when SCM = 1.<br>0 Clock monitor is disabled.<br>1 Clock monitor is enabled. Slow or stopped clocks will cause a clock monitor reset sequence or Self Clock Mode.<br>**Note:** Operating with CME=0 will not detect any loss of clock. In case of poor clock quality this could cause unpredictable operation of the MCU!<br>In Stop Mode (PSTP=0) the clock monitor is disabled independently of the CME bit setting and any loss of external clock will not be detected.<br>Also after wake-up from stop mode (PSTP = 0) with fast wake-up enabled (FSTWKP = 1) the clock monitor is disabled independently of the CME bit setting and any loss of external clock will not be detected. |
| 6<br>PLLON | **Phase Lock Loop On Bit** — PLLON turns on the IPLL circuitry. In Self Clock Mode, the IPLL is turned on, but the PLLON bit reads the last written value. Write anytime except when PLLSEL = 1.<br>0 IPLL is turned off.<br>1 IPLL is turned on. |
| 5, 4<br>FM1, FM0 | IPLL **Frequency Modulation Enable Bit** — FM1 and FM0 enable additional frequency modulation on the VCOCLK. This is to reduce noise emission. The modulation frequency is $f_{ref}$ divided by 16. Write anytime except when PLLSEL = 1. See Table 11-8 for coding. |
| 3<br>FSTWKP | **Fast Wake-up from Full Stop Bit** — FSTWKP enables fast wake-up from full stop mode. Write anytime. If Self-Clock Mode is disabled (SCME = 0) this bit has no effect.<br>0 Fast wake-up from full stop mode is disabled.<br>1 Fast wake-up from full stop mode is enabled. When waking up from full stop mode the system will immediately resume operation in Self-Clock Mode (see Section 11.4.1.4, “Clock Quality Checker”). The SCMIF flag will not be set. The system will remain in Self-Clock Mode with oscillator and clock monitor disabled until FSTWKP bit is cleared. The clearing of FSTWKP will start the oscillator, the clock monitor and the clock quality check. If the clock quality check is successful, the S12XECRG will switch all system clocks to OSCCLK. The SCMIF flag will be set. See application examples in Figure 11-19 and Figure 11-20. |
| 2<br>PRE | **RTI Enable During Pseudo Stop Bit** — PRE enables the RTI during Pseudo Stop Mode.<br>Write anytime.<br>0 RTI stops running during Pseudo Stop Mode.<br>1 RTI continues running during Pseudo Stop Mode.<br>**Note:** If the PRE bit is cleared the RTI dividers will go static while Pseudo Stop Mode is active. The RTI dividers will not initialize like in Wait Mode with RTIWAI bit set. |
| 1<br>PCE | **COP Enable During Pseudo Stop Bit** — PCE enables the COP during Pseudo Stop Mode.<br>Write anytime.<br>0 COP stops running during Pseudo Stop Mode<br>1 COP continues running during Pseudo Stop Mode<br>**Note:** If the PCE bit is cleared the COP dividers will go static while Pseudo Stop Mode is active. The COP dividers will not initialize like in Wait Mode with COPWAI bit set. |
| 0<br>SCME | **Self Clock Mode Enable Bit**<br>Normal modes: Write once<br>Special modes: Write anytime<br>SCME can not be cleared while operating in Self Clock Mode (SCM = 1).<br>0 Detection of crystal clock failure causes clock monitor reset (see Section 11.5.1.1, “Clock Monitor Reset”).<br>1 Detection of crystal clock failure forces the MCU in Self Clock Mode (see Section 11.4.2.2, “Self Clock Mode”). |

**Table 11-8. FM Amplitude selection**

| FM1 | FM0 | FM Amplitude / $f_{VCO}$ Variation |
|---|---|---|
| 0 | 0 | FM off |
| 0 | 1 | ±1% |
| 1 | 0 | ±2% |
| 1 | 1 | ±4% |

### 11.3.2.8 S12XECRG RTI Control Register (RTICTL)

This register selects the timeout period for the Real Time Interrupt.

Module Base + 0x0007

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | RTDEC | RTR6 | RTR5 | RTR4 | RTR3 | RTR2 | RTR1 | RTR0 |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 11-10. S12XECRG RTI Control Register (RTICTL)**

Read: Anytime

Write: Anytime

**NOTE**

A write to this register initializes the RTI counter.

**Table 11-9. RTICTL Field Descriptions**

| Field | Description |
|---|---|
| 7<br>RTDEC | **Decimal or Binary Divider Select Bit** — RTDEC selects decimal or binary based prescaler values.<br>0 Binary based divider value. See Table 11-10<br>1 Decimal based divider value. See Table 11-11 |
| 6–4<br>RTR[6:4] | **Real Time Interrupt Prescale Rate Select Bits** — These bits select the prescale rate for the RTI. See Table 11-10 and Table 11-11. |
| 3–0<br>RTR[3:0] | **Real Time Interrupt Modulus Counter Select Bits** — These bits select the modulus counter target value to provide additional granularity.Table 11-10 and Table 11-11 show all possible divide values selectable by the RTICTL register. The source clock for the RTI is OSCCLK. |

**Table 11-10. RTI Frequency Divide Rates for RTDEC = 0**

| RTR[3:0] | RTR[6:4] = | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | 000 (OFF) | 001 ($2^{10}$) | 010 ($2^{11}$) | 011 ($2^{12}$) | 100 ($2^{13}$) | 101 ($2^{14}$) | 110 ($2^{15}$) | 111 ($2^{16}$) |
| 0000 (÷1) | OFF(1) | $2^{10}$ | $2^{11}$ | $2^{12}$ | $2^{13}$ | $2^{14}$ | $2^{15}$ | $2^{16}$ |

**Table 11-10. RTI Frequency Divide Rates for RTDEC = 0**

| RTR[3:0] | RTR[6:4] = | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | 000 (OFF) | 001 ($2^{10}$) | 010 ($2^{11}$) | 011 ($2^{12}$) | 100 ($2^{13}$) | 101 ($2^{14}$) | 110 ($2^{15}$) | 111 ($2^{16}$) |
| 0001 (÷2) | OFF | $2x2^{10}$ | $2x2^{11}$ | $2x2^{12}$ | $2x2^{13}$ | $2x2^{14}$ | $2x2^{15}$ | $2x2^{16}$ |
| 0010 (÷3) | OFF | $3x2^{10}$ | $3x2^{11}$ | $3x2^{12}$ | $3x2^{13}$ | $3x2^{14}$ | $3x2^{15}$ | $3x2^{16}$ |
| 0011 (÷4) | OFF | $4x2^{10}$ | $4x2^{11}$ | $4x2^{12}$ | $4x2^{13}$ | $4x2^{14}$ | $4x2^{15}$ | $4x2^{16}$ |
| 0100 (÷5) | OFF | $5x2^{10}$ | $5x2^{11}$ | $5x2^{12}$ | $5x2^{13}$ | $5x2^{14}$ | $5x2^{15}$ | $5x2^{16}$ |
| 0101 (÷6) | OFF | $6x2^{10}$ | $6x2^{11}$ | $6x2^{12}$ | $6x2^{13}$ | $6x2^{14}$ | $6x2^{15}$ | $6x2^{16}$ |
| 0110 (÷7) | OFF | $7x2^{10}$ | $7x2^{11}$ | $7x2^{12}$ | $7x2^{13}$ | $7x2^{14}$ | $7x2^{15}$ | $7x2^{16}$ |
| 0111 (÷8) | OFF | $8x2^{10}$ | $8x2^{11}$ | $8x2^{12}$ | $8x2^{13}$ | $8x2^{14}$ | $8x2^{15}$ | $8x2^{16}$ |
| 1000 (÷9) | OFF | $9x2^{10}$ | $9x2^{11}$ | $9x2^{12}$ | $9x2^{13}$ | $9x2^{14}$ | $9x2^{15}$ | $9x2^{16}$ |
| 1001 (÷10) | OFF | $10x2^{10}$ | $10x2^{11}$ | $10x2^{12}$ | $10x2^{13}$ | $10x2^{14}$ | $10x2^{15}$ | $10x2^{16}$ |
| 1010 (÷11) | OFF | $11x2^{10}$ | $11x2^{11}$ | $11x2^{12}$ | $11x2^{13}$ | $11x2^{14}$ | $11x2^{15}$ | $11x2^{16}$ |
| 1011 (÷12) | OFF | $12x2^{10}$ | $12x2^{11}$ | $12x2^{12}$ | $12x2^{13}$ | $12x2^{14}$ | $12x2^{15}$ | $12x2^{16}$ |
| 1100 (÷13) | OFF | $13x2^{10}$ | $13x2^{11}$ | $13x2^{12}$ | $13x2^{13}$ | $13x2^{14}$ | $13x2^{15}$ | $13x2^{16}$ |
| 1101 (÷14) | OFF | $14x2^{10}$ | $14x2^{11}$ | $14x2^{12}$ | $14x2^{13}$ | $14x2^{14}$ | $14x2^{15}$ | $14x2^{16}$ |
| 1110 (÷15) | OFF | $15x2^{10}$ | $15x2^{11}$ | $15x2^{12}$ | $15x2^{13}$ | $15x2^{14}$ | $15x2^{15}$ | $15x2^{16}$ |
| 1111 (÷16) | OFF | $16x2^{10}$ | $16x2^{11}$ | $16x2^{12}$ | $16x2^{13}$ | $16x2^{14}$ | $16x2^{15}$ | $16x2^{16}$ |

1. Denotes the default value out of reset.This value should be used to disable the RTI to ensure future backwards compatibility.

**Table 11-11. RTI Frequency Divide Rates for RTDEC=1**

| RTR[3:0] | RTR[6:4] = | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | 000 ($1x10^3$) | 001 ($2x10^3$) | 010 ($5x10^3$) | 011 ($10x10^3$) | 100 ($20x10^3$) | 101 ($50x10^3$) | 110 ($100x10^3$) | 111 ($200x10^3$) |
| 0000 (÷1) | $1x10^3$ | $2x10^3$ | $5x10^3$ | $10x10^3$ | $20x10^3$ | $50x10^3$ | $100x10^3$ | $200x10^3$ |
| 0001 (÷2) | $2x10^3$ | $4x10^3$ | $10x10^3$ | $20x10^3$ | $40x10^3$ | $100x10^3$ | $200x10^3$ | $400x10^3$ |
| 0010 (÷3) | $3x10^3$ | $6x10^3$ | $15x10^3$ | $30x10^3$ | $60x10^3$ | $150x10^3$ | $300x10^3$ | $600x10^3$ |
| 0011 (÷4) | $4x10^3$ | $8x10^3$ | $20x10^3$ | $40x10^3$ | $80x10^3$ | $200x10^3$ | $400x10^3$ | $800x10^3$ |
| 0100 (÷5) | $5x10^3$ | $10x10^3$ | $25x10^3$ | $50x10^3$ | $100x10^3$ | $250x10^3$ | $500x10^3$ | $1x10^6$ |
| 0101 (÷6) | $6x10^3$ | $12x10^3$ | $30x10^3$ | $60x10^3$ | $120x10^3$ | $300x10^3$ | $600x10^3$ | $1.2x10^6$ |

**Table 11-11. RTI Frequency Divide Rates for RTDEC=1**

| RTR[3:0] | RTR[6:4] = | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | 000 ($1x10^3$) | 001 ($2x10^3$) | 010 ($5x10^3$) | 011 ($10x10^3$) | 100 ($20x10^3$) | 101 ($50x10^3$) | 110 ($100x10^3$) | 111 ($200x10^3$) |
| **0110 (÷7)** | $7x10^3$ | $14x10^3$ | $35x10^3$ | $70x10^3$ | $140x10^3$ | $350x10^3$ | $700x10^3$ | $1.4x10^6$ |
| **0111 (÷8)** | $8x10^3$ | $16x10^3$ | $40x10^3$ | $80x10^3$ | $160x10^3$ | $400x10^3$ | $800x10^3$ | $1.6x10^6$ |
| **1000 (÷9)** | $9x10^3$ | $18x10^3$ | $45x10^3$ | $90x10^3$ | $180x10^3$ | $450x10^3$ | $900x10^3$ | $1.8x10^6$ |
| **1001 (÷10)** | $10 x10^3$ | $20x10^3$ | $50x10^3$ | $100x10^3$ | $200x10^3$ | $500x10^3$ | $1x10^6$ | $2x10^6$ |
| **1010 (÷11)** | $11 x10^3$ | $22x10^3$ | $55x10^3$ | $110x10^3$ | $220x10^3$ | $550x10^3$ | $1.1x10^6$ | $2.2x10^6$ |
| **1011 (÷12)** | $12x10^3$ | $24x10^3$ | $60x10^3$ | $120x10^3$ | $240x10^3$ | $600x10^3$ | $1.2x10^6$ | $2.4x10^6$ |
| **1100 (÷13)** | $13x10^3$ | $26x10^3$ | $65x10^3$ | $130x10^3$ | $260x10^3$ | $650x10^3$ | $1.3x10^6$ | $2.6x10^6$ |
| **1101 (÷14)** | $14x10^3$ | $28x10^3$ | $70x10^3$ | $140x10^3$ | $280x10^3$ | $700x10^3$ | $1.4x10^6$ | $2.8x10^6$ |
| **1110 (÷15)** | $15x10^3$ | $30x10^3$ | $75x10^3$ | $150x10^3$ | $300x10^3$ | $750x10^3$ | $1.5x10^6$ | $3x10^6$ |
| **1111 (÷16)** | $16x10^3$ | $32x10^3$ | $80x10^3$ | $160x10^3$ | $320x10^3$ | $800x10^3$ | $1.6x10^6$ | $3.2x10^6$ |

### 11.3.2.9 S12XECRG COP Control Register (COPCTL)

This register controls the COP (Computer Operating Properly) watchdog.

Module Base + 0x0008

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | WCOP | RSBCK | 0 | 0 | 0 | CR2 | CR1 | CR0 |
| W | | | WRTMASK | | | | | |
| Reset[1] | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

1. Refer to Device User Guide (Section: S12XECRG) for reset values of WCOP, CR2, CR1 and CR0.

= Unimplemented or Reserved

**Figure 11-11. S12XECRG COP Control Register (COPCTL)**

Read: Anytime

Write:

1. RSBCK: anytime in special modes; write to “1” but not to “0” in all other modes
2. WCOP, CR2, CR1, CR0:
   — Anytime in special modes
   — Write once in all other modes
     – Writing CR[2:0] to “000” has no effect, but counts for the “write once” condition.
     – Writing WCOP to “0” has no effect, but counts for the “write once” condition.

The COP time-out period is restarted if one these two conditions is true:

1. Writing a non zero value to CR[2:0] (anytime in special modes, once in all other modes) with WRTMASK = 0.

   or

2. Changing RSBCK bit from “0” to “1”.

**Table 11-12. COPCTL Field Descriptions**

| Field | Description |
|---|---|
| 7<br>WCOP | **Window COP Mode Bit** — When set, a write to the ARMCOP register must occur in the last 25% of the selected period. A write during the first 75% of the selected period will reset the part. As long as all writes occur during this window, $55 can be written as often as desired. Once $AA is written after the $55, the time-out logic restarts and the user must wait until the next window before writing to ARMCOP. Table 11-13 shows the duration of this window for the seven available COP rates.<br>0 Normal COP operation<br>1 Window COP operation |
| 6<br>RSBCK | **COP and RTI Stop in Active BDM Mode Bit**<br>0 Allows the COP and RTI to keep running in Active BDM mode.<br>1 Stops the COP and RTI counters whenever the part is in Active BDM mode. |
| 5<br>WRTMASK | **Write Mask for WCOP and CR[2:0] Bit** — This write-only bit serves as a mask for the WCOP and CR[2:0] bits while writing the COPCTL register. It is intended for BDM writing the RSBCK without touching the contents of WCOP and CR[2:0].<br>0 Write of WCOP and CR[2:0] has an effect with this write of COPCTL<br>1 Write of WCOP and CR[2:0] has no effect with this write of COPCTL.<br>(Does not count for “write once”.) |
| 2–0<br>CR[2:0] | **COP Watchdog Timer Rate Select** — These bits select the COP time-out rate (see Table 11-13). Writing a nonzero value to CR[2:0] enables the COP counter and starts the time-out period. A COP counter time-out causes a system reset. This can be avoided by periodically (before time-out) reinitialize the COP counter via the ARMCOP register.<br>While all of the following four conditions are true the CR[2:0], WCOP bits are ignored and the COP operates at highest time-out period ($2^{24}$ cycles) in normal COP mode (Window COP mode disabled):<br>1) COP is enabled (CR[2:0] is not 000)<br>2) BDM mode active<br>3) RSBCK = 0<br>4) Operation in emulation or special modes |

**Table 11-13. COP Watchdog Rates(1)**

| CR2 | CR1 | CR0 | OSCCLK Cycles to Timeout |
|---|---|---|---|
| 0 | 0 | 0 | COP disabled |
| 0 | 0 | 1 | $2^{14}$ |
| 0 | 1 | 0 | $2^{16}$ |
| 0 | 1 | 1 | $2^{18}$ |
| 1 | 0 | 0 | $2^{20}$ |
| 1 | 0 | 1 | $2^{22}$ |
| 1 | 1 | 0 | $2^{23}$ |

**Table 11-13. COP Watchdog Rates(1)**

| CR2 | CR1 | CR0 | OSCCLK Cycles to Timeout |
|---|---|---|---|
| 1 | 1 | 1 | $2^{24}$ |

1. OSCCLK cycles are referenced from the previous COP time-out reset (writing $55/$AA to the ARMCOP register)

### 11.3.2.10 Reserved Register (FORBYP)

**NOTE**

This reserved register is designed for factory test purposes only, and is not intended for general user access. Writing to this register when in special modes can alter the S12XECRG's functionality.

Module Base + 0x0009

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| W | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented or Reserved

**Figure 11-12. Reserved Register (FORBYP)**

Read: Always read $00 except in special modes

Write: Only in special modes

### 11.3.2.11 Reserved Register (CTCTL)

**NOTE**

This reserved register is designed for factory test purposes only, and is not intended for general user access. Writing to this register when in special test modes can alter the S12XECRG's functionality.

Module Base + 0x000A

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| W | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented or Reserved

**Figure 11-13. Reserved Register (CTCTL)**

Read: Always read $00 except in special modes

Write: Only in special modes

### 11.3.2.12 S12XECRG COP Timer Arm/Reset Register (ARMCOP)

This register is used to restart the COP time-out period.

Module Base + 0x000B

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| W | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 11-14. S12XECRG ARMCOP Register Diagram**

Read: Always reads $00

Write: Anytime

When the COP is disabled (CR[2:0] = “000”) writing to this register has no effect.

When the COP is enabled by setting CR[2:0] nonzero, the following applies:

Writing any value other than $55 or $AA causes a COP reset. To restart the COP time-out period you must write $55 followed by a write of $AA. Other instructions may be executed between these writes but the sequence ($55, $AA) must be completed prior to COP end of time-out period to avoid a COP reset. Sequences of $55 writes or sequences of $AA writes are allowed. When the WCOP bit is set, $55 and $AA writes must be done in the last 25% of the selected time-out period; writing any value in the first 75% of the selected period will cause a COP reset.

# 11.4 Functional Description

## 11.4.1 Functional Blocks

### 11.4.1.1 Phase Locked Loop with Internal Filter (IPLL)

The IPLL is used to run the MCU from a different time base than the incoming OSCCLK. Figure 11-15 shows a block diagram of the IPLL.

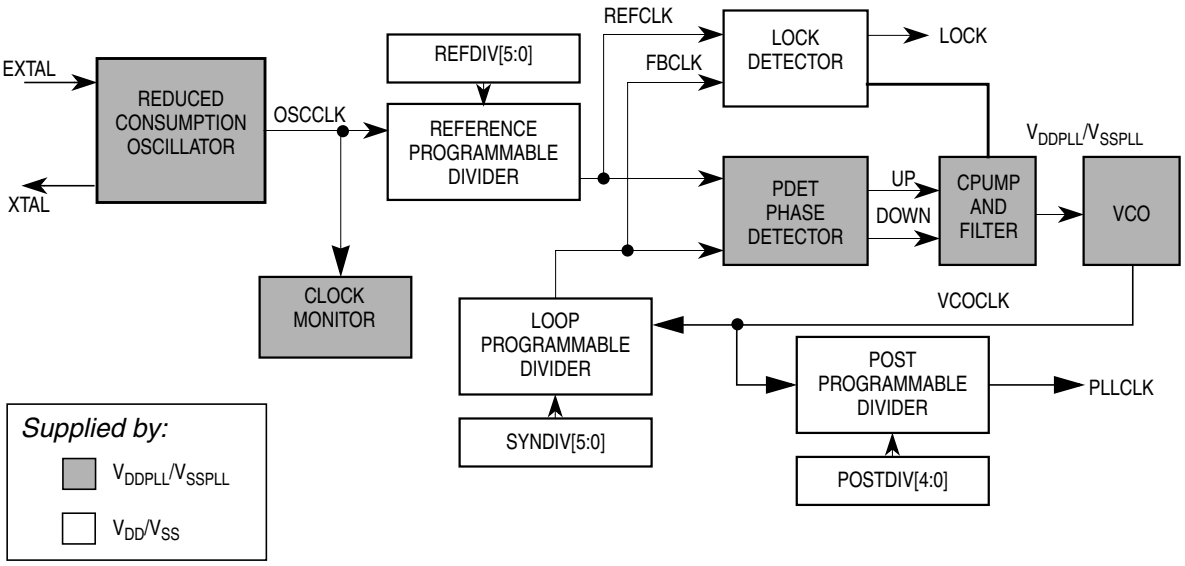



**Figure 11-15. IPLL Functional Diagram**

For increased flexibility, OSCCLK can be divided in a range of 1 to 64 to generate the reference frequency REFCLK using the REFDIV[5:0] bits. This offers a finer multiplication granularity. Based on the SYNDIV[5:0] bits the IPLL generates the VCOCLK by multiplying the reference clock by a multiple of 2, 4, 6,... 126, 128. Based on the POSTDIV[4:0] bits the VCOCLK can be divided in a range of 1,2,4,6,8,... to 62 to generate the PLLCLK.

$$f_{PLL} = 2 \times f_{OSC} \times \frac{SYNDIV + 1}{[REFDIV + 1][2 \times POSTDIV]}$$

**NOTE**

Although it is possible to set the dividers to command a very high clock frequency, do not exceed the specified bus frequency limit for the MCU.

If (PLLSEL = 1) then $f_{BUS} = f_{PLL} / 2$.

IF POSTDIV = $00 the $f_{PLL}$ is identical to $f_{VCO}$ (divide by one)

Several examples of IPLL divider settings are shown in Table 11-14. Shaded rows indicated that these settings are not recommended. The following rules help to achieve optimum stability and shortest lock time:

- Use lowest possible $f_{VCO} / f_{REF}$ ratio (SYNDIV value).
- Use highest possible REFCLK frequency $f_{REF}$.

**Table 11-14. Examples of IPLL Divider Settings[1]**

| $f_{OSC}$ | REFDIV[5:0] | $f_{REF}$ | REFFRQ[1:0] | SYNDIV[5:0] | $f_{VCO}$ | VCOFRQ[1:0] | POSTDIV[4:0] | $f_{PLL}$ | $f_{BUS}$ |
|---|---|---|---|---|---|---|---|---|---|
| 4MHz | $01 | 2MHz | 01 | $18 | 100MHz | 11 | $00 | 100MHz | 50 MHz |
| 8MHz | $03 | 2MHz | 01 | $18 | 100MHz | 11 | $00 | 100MHz | 50 MHz |
| 4MHz | $00 | 4MHz | 01 | $09 | 80MHz | 01 | $00 | 80MHz | 40MHz |
| 8MHz | $00 | 8MHz | 10 | $04 | 80MHz | 01 | $00 | 80MHz | 40MHz |
| 4MHz | $00 | 4MHz | 01 | $03 | 32MHz | 00 | $01 | 16MHz | 8MHz |
| 4MHz | $01 | 2MHz | 01 | $18 | 100MHz | 11 | $01 | 50MHz | 25MHz |

1. $f_{PLL}$ and $f_{BUS}$ values in this table may exceed maximum allowed frequencies for some devices. Refer to device information for maximum values.

#### 11.4.1.1.1 IPLL Operation

The oscillator output clock signal (OSCCLK) is fed through the reference programmable divider and is divided in a range of 1 to 64 (REFDIV+1) to output the REFCLK. The VCO output clock, (VCOCLK) is fed back through the programmable loop divider and is divided in a range of 2 to 128 in increments of [2 x (SYNDIV +1)] to output the FBCLK. The VCOCLK is fed to the final programmable divider and is divided in a range of 1,2,4,6,8,... to 62 (2*POSTDIV) to output the PLLCLK. See Figure 11-15.

The phase detector then compares the FBCLK, with the REFCLK. Correction pulses are generated based on the phase difference between the two signals. The loop filter then slightly alters the DC voltage on the internal filter capacitor, based on the width and direction of the correction pulse.

The user must select the range of the REFCLK frequency and the range of the VCOCLK frequency to ensure that the correct IPLL loop bandwidth is set.

The lock detector compares the frequencies of the FBCLK, and the REFCLK. Therefore, the speed of the lock detector is directly proportional to the reference clock frequency. The circuit determines the lock condition based on this comparison.

If IPLL LOCK interrupt requests are enabled, the software can wait for an interrupt request and then check the LOCK bit. If interrupt requests are disabled, software can poll the LOCK bit continuously (during IPLL start-up, usually) or at periodic intervals. In either case, only when the LOCK bit is set, the PLLCLK can be selected as the source for the system and core clocks. If the IPLL is selected as the source for the system and core clocks and the LOCK bit is clear, the IPLL has suffered a severe noise hit and the software must take appropriate action, depending on the application.

- The LOCK bit is a read-only indicator of the locked state of the IPLL.
- The LOCK bit is set when the VCO frequency is within a certain tolerance, $\Delta_{Lock}$, and is cleared when the VCO frequency is out of a certain tolerance, $\Delta_{unl}$.
- Interrupt requests can occur if enabled (LOCKIE = 1) when the lock condition changes, toggling the LOCK bit.

### 11.4.1.2 System Clocks Generator

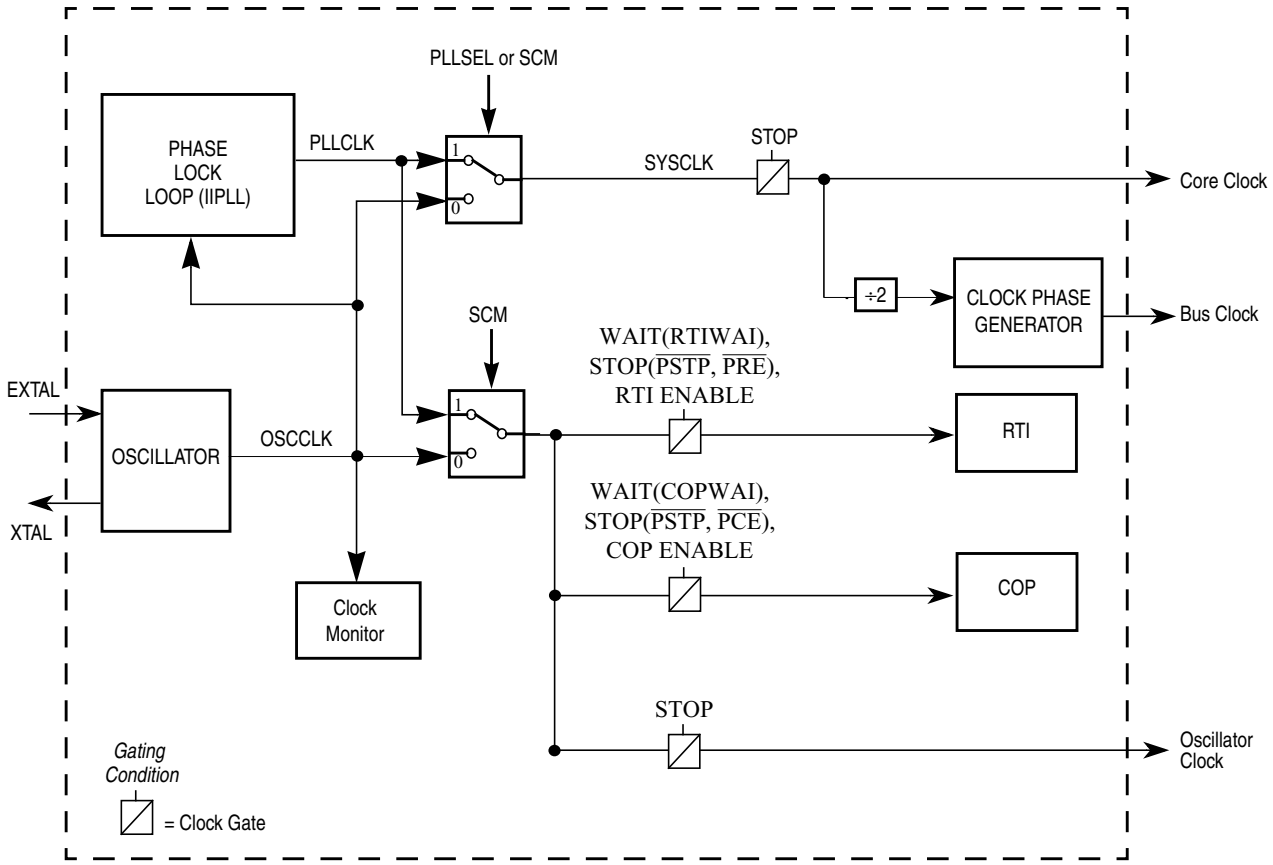



**Figure 11-16. System Clocks Generator**

The clock generator creates the clocks used in the MCU (see Figure 11-16). The gating condition placed on top of the individual clock gates indicates the dependencies of different modes (STOP, WAIT) and the setting of the respective configuration bits.

The peripheral modules use the Bus Clock. Some peripheral modules also use the Oscillator Clock. If the MCU enters Self Clock Mode (see Section 11.4.2.2, “Self Clock Mode”) Oscillator clock source is switched to PLLCLK running at its minimum frequency $f_{SCM}$. The Bus Clock is used to generate the clock visible at the ECLK pin. The Core Clock signal is the clock for the CPU. The Core Clock is twice the Bus Clock. But note that a CPU cycle corresponds to one Bus Clock.

IPLL clock mode is selected with PLLSEL bit in the CLKSEL register. When selected, the IPLL output clock drives SYSCLK for the main system including the CPU and peripherals. The IPLL cannot be turned off by clearing the PLLON bit, if the IPLL clock is selected. When PLLSEL is changed, it takes a maximum of 4 OSCCLK plus 4 PLLCLK cycles to make the transition. During the transition, all clocks freeze and CPU activity ceases.

#### 11.4.1.3 Clock Monitor (CM)

If no OSCCLK edges are detected within a certain time, the clock monitor within the oscillator block generates a clock monitor fail event. The S12XECRG then asserts self clock mode or generates a system reset depending on the state of SCME bit. If the clock monitor is disabled or the presence of clocks is detected no failure is indicated by the oscillator block.The clock monitor function is enabled/disabled by the CME control bit.

#### 11.4.1.4 Clock Quality Checker

The clock monitor performs a coarse check on the incoming clock signal. The clock quality checker provides a more accurate check in addition to the clock monitor.

A clock quality check is triggered by any of the following events:

- Power on reset (*POR*)
- Low voltage reset (*LVR*)
- Wake-up from Full Stop Mode (*exit full stop*)
- Clock Monitor fail indication (*CM fail*)

A time window of 50000 PLLCLK cycles[1] is called *check window*.

A number greater equal than 4096 rising OSCCLK edges within a *check window* is called *osc ok*. Note that *osc ok* immediately terminates the current *check window*. See Figure 11-17 as an example.

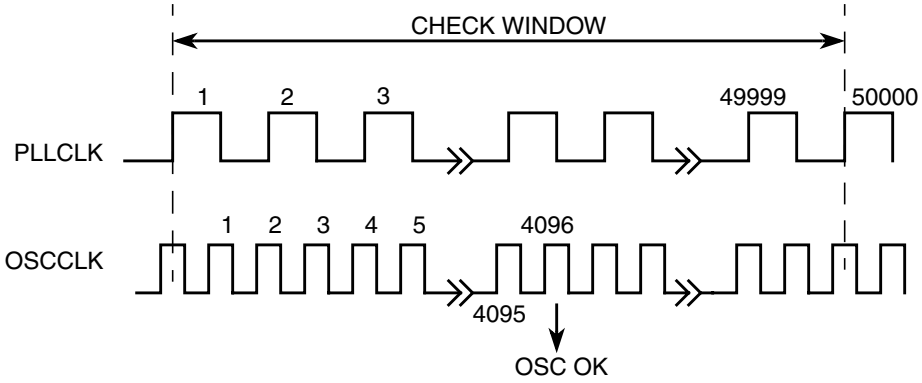



**Figure 11-17. Check Window Example**

1. IPLL is running at self clock mode frequency $f_{SCM}$.

The Sequence for clock quality check is shown in Figure 11-18.

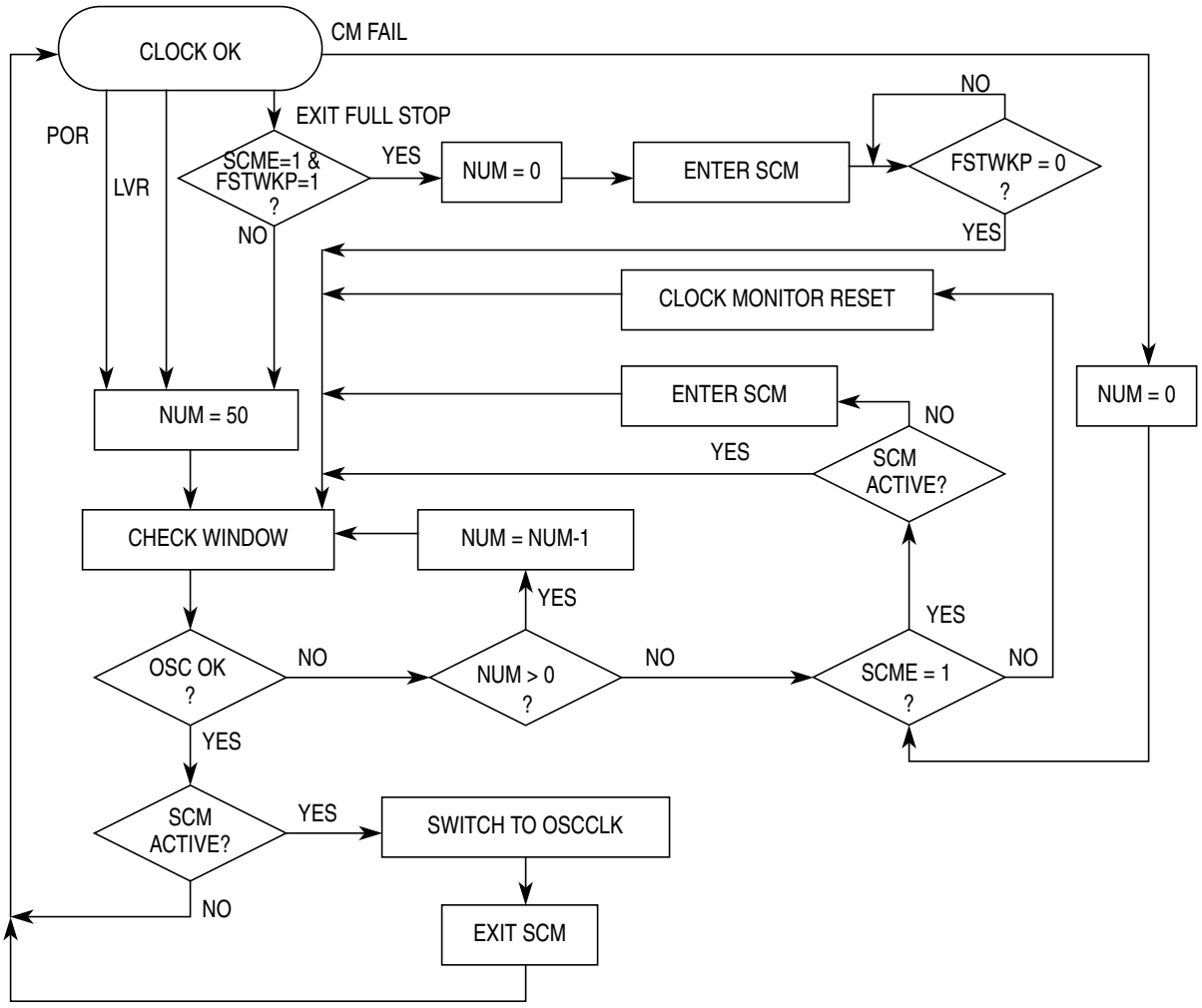



**Figure 11-18. Sequence for Clock Quality Check**

## NOTE

Remember that in parallel to additional actions caused by Self Clock Mode or Clock Monitor Reset[1] handling the clock quality checker **continues** to check the OSCCLK signal.

## NOTE

The Clock Quality Checker enables the IPLL and the voltage regulator (VREG) anytime a clock check has to be performed. An ongoing clock quality check could also cause a running IPLL ($f_{SCM}$) and an active VREG during Pseudo Stop Mode.

1. A Clock Monitor Reset will always set the SCME bit to logical'1'.

#### 11.4.1.5 Computer Operating Properly Watchdog (COP)

The COP (free running watchdog timer) enables the user to check that a program is running and sequencing properly. When the COP is being used, software is responsible for keeping the COP from timing out. If the COP times out it is an indication that the software is no longer being executed in the intended sequence; thus a system reset is initiated (see Section 11.4.1.5, “Computer Operating Properly Watchdog (COP)”). The COP runs with a gated OSCCLK. Three control bits in the COPCTL register allow selection of seven COP time-out periods.

When COP is enabled, the program must write $55 and $AA (in this order) to the ARMCOP register during the selected time-out period. Once this is done, the COP time-out period is restarted. If the program fails to do this and the COP times out, the part will reset. Also, if any value other than $55 or $AA is written, the part is immediately reset.

Windowed COP operation is enabled by setting WCOP in the COPCTL register. In this mode, writes to the ARMCOP register to clear the COP timer must occur in the last 25% of the selected time-out period. A premature write will immediately reset the part.

If PCE bit is set, the COP will continue to run in Pseudo Stop Mode.

#### 11.4.1.6 Real Time Interrupt (RTI)

The RTI can be used to generate a hardware interrupt at a fixed periodic rate. If enabled (by setting RTIE=1), this interrupt will occur at the rate selected by the RTICTL register. The RTI runs with a gated OSCCLK. At the end of the RTI time-out period the RTIF flag is set to one and a new RTI time-out period starts immediately.

A write to the RTICTL register restarts the RTI time-out period.

If the PRE bit is set, the RTI will continue to run in Pseudo Stop Mode.

### 11.4.2 Operation Modes

#### 11.4.2.1 Normal Mode

The S12XECRG block behaves as described within this specification in all normal modes.

#### 11.4.2.2 Self Clock Mode

If the external clock frequency is not available due to a failure or due to long crystal start-up time, the Bus Clock and the Core Clock are derived from the PLLCLK running at self clock mode frequency $f_{SCM}$; this mode of operation is called Self Clock Mode. This requires CME = 1 and SCME = 1, which is the default after reset. If the MCU was clocked by the PLLCLK prior to entering Self Clock Mode, the PLLSEL bit will be cleared. If the external clock signal has stabilized again, the S12XECRG will automatically select OSCCLK to be the system clock and return to normal mode. See Section 11.4.1.4, “Clock Quality Checker” for more information on entering and leaving Self Clock Mode.

**NOTE**

In order to detect a potential clock loss the CME bit should always be enabled (CME = 1).

If CME bit is disabled and the MCU is configured to run on PLLCLK, a loss of external clock (OSCCLK) will not be detected and will cause the system clock to drift towards lower frequencies. As soon as the external clock is available again the system clock ramps up to its IPLL target frequency. If the MCU is running on external clock any loss of clock will cause the system to go static.

## 11.4.3 Low Power Options

This section summarizes the low power options available in the S12XECRG.

### 11.4.3.1 Run Mode

This is the default mode after reset.

The RTI can be stopped by setting the associated rate select bits to zero.

The COP can be stopped by setting the associated rate select bits to zero.

### 11.4.3.2 Wait Mode

The WAI instruction puts the MCU in a low power consumption stand-by mode depending on setting of the individual bits in the CLKSEL register. All individual Wait Mode configuration bits can be superposed. This provides enhanced granularity in reducing the level of power consumption during Wait Mode. Table 11-15 lists the individual configuration bits and the parts of the MCU that are affected in Wait Mode.

**Table 11-15. MCU Configuration During Wait Mode**

| | PLLWAI | RTIWAI | COPWAI |
|---|---|---|---|
| **IPLL** | Stopped | — | — |
| **RTI** | — | Stopped | — |
| **COP** | — | — | Stopped |

After executing the WAI instruction the core requests the S12XECRG to switch MCU into Wait Mode. The S12XECRG then checks whether the PLLWAI bit is asserted. Depending on the configuration the S12XECRG switches the system and core clocks to OSCCLK by clearing the PLLSEL bit and disables the IPLL.

There are two ways to restart the MCU from Wait Mode:

1. Any reset
2. Any interrupt

#### 11.4.3.3 Stop Mode

All clocks are stopped in STOP mode, dependent of the setting of the PCE, PRE and PSTP bit. The oscillator is disabled in STOP mode unless the PSTP bit is set. If the PRE or PCE bits are set, the RTI or COP continues to run in Pseudo Stop Mode. In addition to disabling system and core clocks the S12XECRG requests other functional units of the MCU (e.g. voltage-regulator) to enter their individual power saving modes (if available).

If the PLLSEL bit is still set when entering Stop Mode, the S12XECRG will switch the system and core clocks to OSCCLK by clearing the PLLSEL bit. Then the S12XECRG disables the IPLL, disables the core clock and finally disables the remaining system clocks.

If Pseudo Stop Mode is entered from Self-Clock Mode the S12XECRG will continue to check the clock quality until clock check is successful. In this case the IPLL and the voltage regulator (VREG) will remain enabled. If Full Stop Mode (PSTP = 0) is entered from Self-Clock Mode the ongoing clock quality check will be stopped. A complete timeout window check will be started when Stop Mode is left again.

There are two ways to restart the MCU from Stop Mode:

1. Any reset
2. Any interrupt

If the MCU is woken-up from Full Stop Mode by an interrupt and the fast wake-up feature is enabled (FSTWKP=1 and SCME=1), the system will immediately (no clock quality check) resume operation in Self-Clock Mode (see Section 11.4.1.4, “Clock Quality Checker”). The SCMIF flag will not be set for this special case. The system will remain in Self-Clock Mode with oscillator disabled until FSTWKP bit is cleared. The clearing of FSTWKP will start the oscillator and the clock quality check. If the clock quality check is successful, the S12XECRG will switch all system clocks to oscillator clock. The SCMIF flag will be set. See application examples in Figure 11-19 and Figure 11-20.

Because the IPLL has been powered-down during Stop Mode the PLLSEL bit is cleared and the MCU runs on OSCCLK after leaving Stop-Mode. The software must manually set the PLLSEL bit again, in order to switch system and core clocks to the PLLCLK.

**NOTE**

In Full Stop Mode or Self-Clock Mode caused by the fast wake-up feature the clock monitor and the oscillator are disabled.

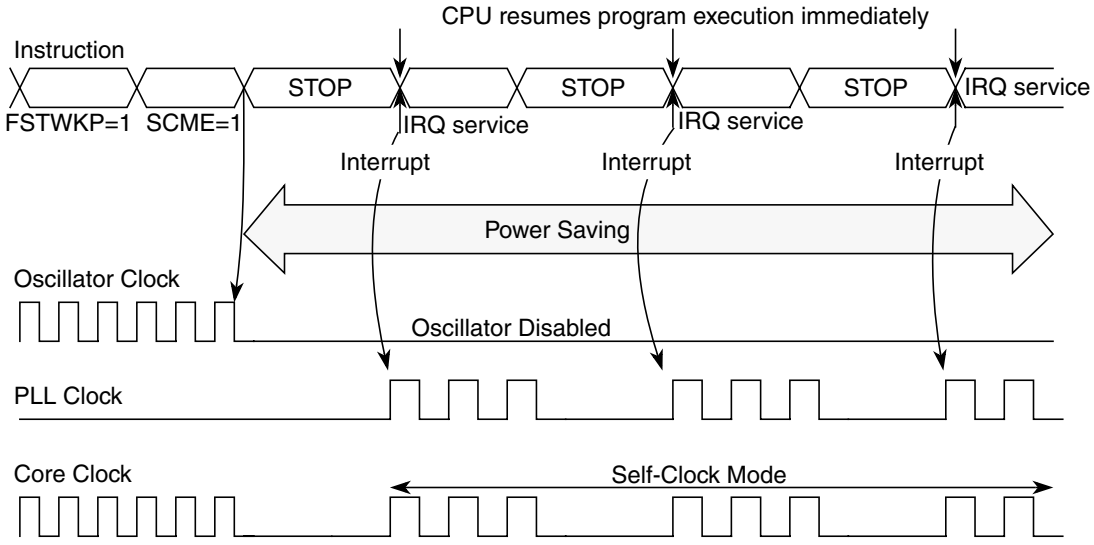



**Figure 11-19. Fast Wake-up from Full Stop Mode: Example 1**

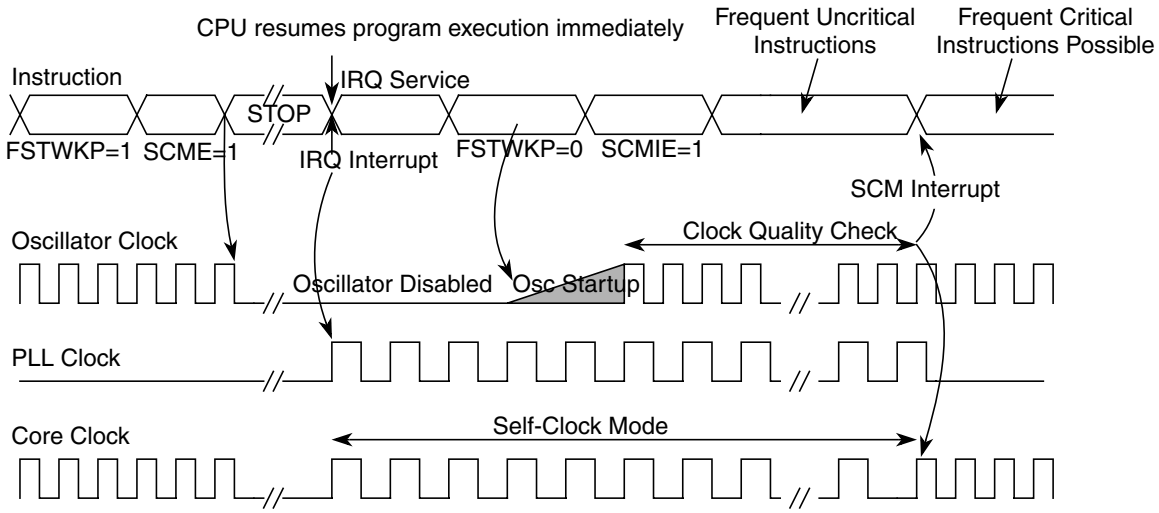



**Figure 11-20. Fast Wake-up from Full Stop Mode: Example 2**

## 11.5 Resets

All reset sources are listed in Table 11-16. Refer to MCU specification for related vector addresses and priorities.

**Table 11-16. Reset Summary**

| Reset Source | Local Enable |
|---|---|
| Power on Reset | None |
| Low Voltage Reset | None |
| External Reset | None |
| Illegal Address Reset | None |
| Clock Monitor Reset | PLLCTL (CME=1, SCME=0) |

**Table 11-16. Reset Summary**

| Reset Source | Local Enable |
|---|---|
| COP Watchdog Reset | COPCTL (CR[2:0] nonzero) |

## 11.5.1 Description of Reset Operation

The reset sequence is initiated by any of the following events:

- Low level is detected at the $\overline{\text{RESET}}$ pin (External Reset).
- Power on is detected.
- Low voltage is detected.
- Illegal Address Reset is detected (refer to device MMC information for details).
- COP watchdog times out.
- Clock monitor failure is detected and Self-Clock Mode was disabled (SCME=0).

Upon detection of any reset event, an internal circuit drives the $\overline{\text{RESET}}$ pin low for 128 SYSCLK cycles (see Figure 11-21). Since entry into reset is asynchronous it does not require a running SYSCLK. However, the internal reset circuit of the S12XECRG cannot sequence out of current reset condition without a running SYSCLK. The number of 128 SYSCLK cycles might be increased by n = 3 to 6 additional SYSCLK cycles depending on the internal synchronization latency. After 128+n SYSCLK cycles the $\overline{\text{RESET}}$ pin is released. The reset generator of the S12XECRG waits for additional 64 SYSCLK cycles and then samples the RESET pin to determine the originating source. Table 11-17 shows which vector will be fetched.

**Table 11-17. Reset Vector Selection**

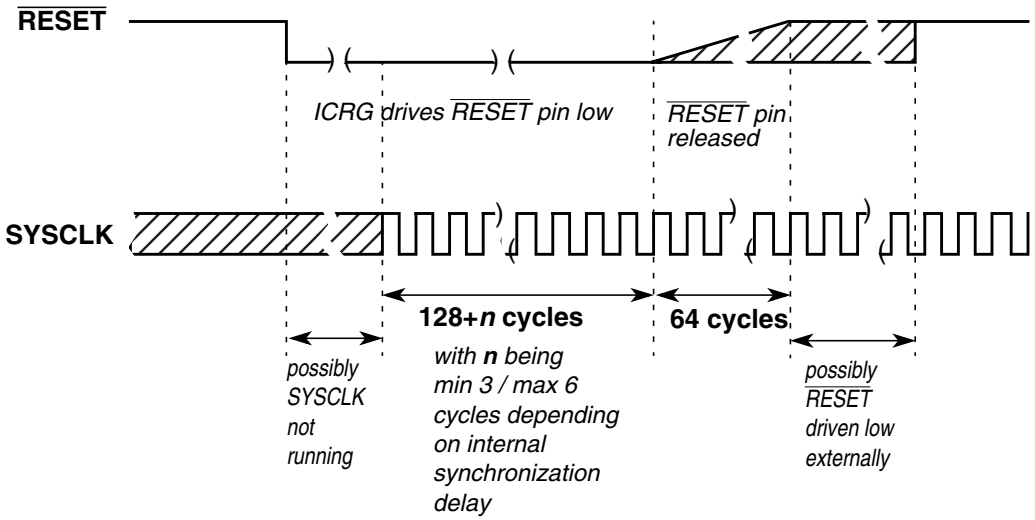
| Sampled $\overline{\text{RESET}}$ Pin (64 cycles after release) | Clock Monitor Reset Pending | COP Reset Pending | Vector Fetch |
|---|---|---|---|
| 1 | 0 | 0 | POR / LVR / Illegal Address Reset/ External Reset |
| 1 | 1 | X | Clock Monitor Reset |
| 1 | 0 | 1 | COP Reset |
| 0 | X | X | POR / LVR / Illegal Address Reset/ External Reset with rise of $\overline{\text{RESET}}$ pin |

**NOTE**

External circuitry connected to the RESET pin should be able to raise the signal to a valid logic one within 64 SYSCLK cycles after the low drive is released by the MCU. If this requirement is not adhered to the reset source will always be recognized as “External Reset” even if the reset was initially caused by an other reset source.

The internal reset of the MCU remains asserted while the reset generator completes the 192 SYSCLK long reset sequence. In case the RESET pin is externally driven low for more than these 192 SYSCLK cycles (External Reset), the internal reset remains asserted longer.

**Figure 11-21. RESET Timing**




### 11.5.1.1 Clock Monitor Reset

The S12XECRG generates a Clock Monitor Reset in case all of the following conditions are true:

- Clock monitor is enabled (CME = 1)
- Loss of clock is detected
- Self-Clock Mode is disabled (SCME = 0).

The reset event asynchronously forces the configuration registers to their default settings. In detail the CME and the SCME are reset to logical ‘1’ (which changes the state of the SCME bit. As a consequence the S12XECRG immediately enters Self Clock Mode and starts its internal reset sequence. In parallel the clock quality check starts. As soon as clock quality check indicates a valid Oscillator Clock the S12XECRG switches to OSCCLK and leaves Self Clock Mode. Since the clock quality checker is running in parallel to the reset generator, the S12XECRG may leave Self Clock Mode while still completing the internal reset sequence.

### 11.5.1.2 Computer Operating Properly Watchdog (COP) Reset

When COP is enabled, the S12XECRG expects sequential write of $55 and $AA (in this order) to the ARMCOP register during the selected time-out period. Once this is done, the COP time-out period restarts. If the program fails to do this the S12XECRG will generate a reset.

### 11.5.1.3 Power On Reset, Low Voltage Reset

The on-chip voltage regulator detects when $V_{DD}$ to the MCU has reached a certain level and asserts power on reset or low voltage reset or both. As soon as a power on reset or low voltage reset is triggered the

S12XECRG performs a quality check on the incoming clock signal. As soon as clock quality check indicates a valid Oscillator Clock signal the reset sequence starts using the Oscillator clock. If after 50 check windows the clock quality check indicated a non-valid Oscillator Clock the reset sequence starts using Self-Clock Mode.

Figure 11-22 and Figure 11-23 show the power-up sequence for cases when the $\overline{\text{RESET}}$ pin is tied to $V_{DD}$ and when the $\overline{\text{RESET}}$ pin is held low.

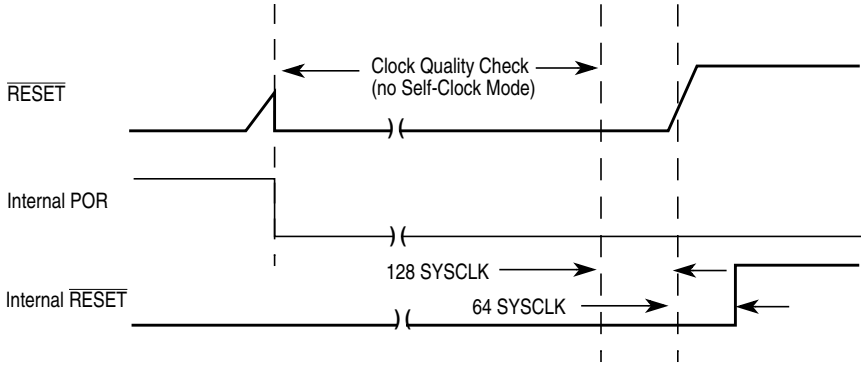



**Figure 11-22. $\overline{\text{RESET}}$ Pin Tied to $V_{DD}$ (by a Pull-up Resistor)**

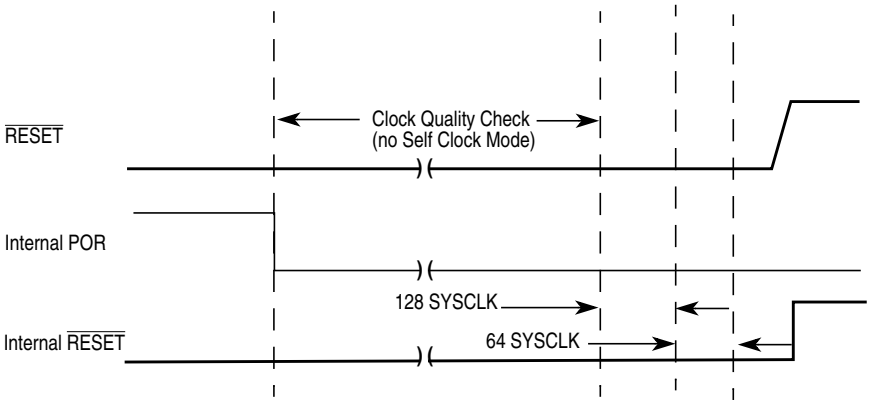



**Figure 11-23. $\overline{\text{RESET}}$ Pin Held Low Externally**

## 11.6 Interrupts

The interrupts/reset vectors requested by the S12XECRG are listed in Table 11-18. Refer to MCU specification for related vector addresses and priorities.

**Table 11-18. S12XECRG Interrupt Vectors**

| Interrupt Source | CCR Mask | Local Enable |
|---|---|---|
| Real time interrupt | I bit | CRGINT (RTIE) |
| LOCK interrupt | I bit | CRGINT (LOCKIE) |
| SCM interrupt | I bit | CRGINT (SCMIE) |

## 11.6.1 Description of Interrupt Operation

### 11.6.1.1 Real Time Interrupt

The S12XECRG generates a real time interrupt when the selected interrupt time period elapses. RTI interrupts are locally disabled by setting the RTIE bit to zero. The real time interrupt flag (RTIF) is set to1 when a timeout occurs, and is cleared to 0 by writing a 1 to the RTIF bit.

The RTI continues to run during Pseudo Stop Mode if the PRE bit is set to 1. This feature can be used for periodic wakeup from Pseudo Stop if the RTI interrupt is enabled.

### 11.6.1.2 IPLL Lock Interrupt

The S12XECRG generates a IPLL Lock interrupt when the LOCK condition of the IPLL has changed, either from a locked state to an unlocked state or vice versa. Lock interrupts are locally disabled by setting the LOCKIE bit to zero. The IPLL Lock interrupt flag (LOCKIF) is set to1 when the LOCK condition has changed, and is cleared to 0 by writing a 1 to the LOCKIF bit.

### 11.6.1.3 Self Clock Mode Interrupt

The S12XECRG generates a Self Clock Mode interrupt when the SCM condition of the system has changed, either entered or exited Self Clock Mode. SCM conditions are caused by a failing clock quality check after power on reset (POR) or low voltage reset (LVR) or recovery from Full Stop Mode (PSTP = 0) or Clock Monitor failure. For details on the clock quality check refer to Section 11.4.1.4, “Clock Quality Checker”. If the clock monitor is enabled (CME = 1) a loss of external clock will also cause a SCM condition (SCME = 1).

SCM interrupts are locally disabled by setting the SCMIE bit to zero. The SCM interrupt flag (SCMIF) is set to1 when the SCM condition has changed, and is cleared to 0 by writing a 1 to the SCMIF bit.

# Chapter 12
# Pierce Oscillator (S12XOSCLCPV2)

**Table 12-1. Revision History**

| Revision Number | Revision Date | Sections Affected | Description of Changes |
|---|---|---|---|
| V01.05 | 19 Jul 2006 | | - All xclks info was removed |
| V02.00 | 04 Aug 2006 | | - Incremented revision to match the design system spec revision |

## 12.1 Introduction

The Pierce oscillator (XOSC) module provides a robust, low-noise and low-power clock source. The module will be operated from the $V_{DDPLL}$ supply rail (1.8 V nominal) and require the minimum number of external components. It is designed for optimal start-up margin with typical crystal oscillators.

### 12.1.1 Features

The XOSC will contain circuitry to dynamically control current gain in the output amplitude. This ensures a signal with low harmonic distortion, low power and good noise immunity.

- High noise immunity due to input hysteresis
- Low RF emissions with peak-to-peak swing limited dynamically
- Transconductance (gm) sized for optimum start-up margin for typical oscillators
- Dynamic gain control eliminates the need for external current limiting resistor
- Integrated resistor eliminates the need for external bias resistor in loop controlled Pierce mode.
- Low power consumption:
  - — Operates from 1.8 V (nominal) supply
  - — Amplitude control limits power
- Clock monitor

### 12.1.2 Modes of Operation

Two modes of operation exist:

1. Loop controlled Pierce (LCP) oscillator
2. External square wave mode featuring also full swing Pierce (FSP) without internal bias resistor

The oscillator mode selection is described in the Device Overview section, subsection Oscillator Configuration.

### 12.1.3 Block Diagram

Figure 12-1 shows a block diagram of the XOSC.

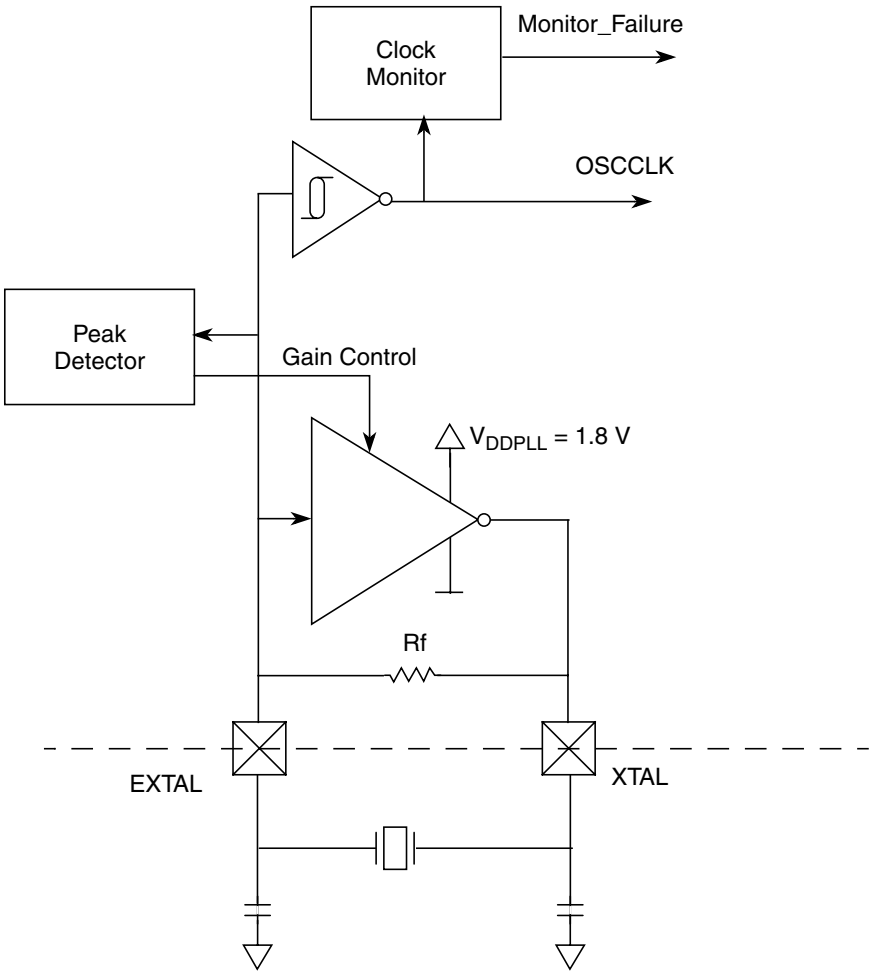



**Figure 12-1. XOSC Block Diagram**

## 12.2 External Signal Description

This section lists and describes the signals that connect off chip

### 12.2.1 VDDPLL and VSSPLL — Operating and Ground Voltage Pins

Theses pins provides operating voltage ($V_{DDPLL}$) and ground ($V_{SSPLL}$) for the XOSC circuitry. This allows the supply voltage to the XOSC to use an independent bypass capacitor.

### 12.2.2 EXTAL and XTAL — Input and Output Pins

These pins provide the interface for either a crystal or a 1.8V CMOS compatible clock to control the internal clock generator circuitry. EXTAL is the external clock input or the input to the crystal oscillator amplifier. XTAL is the output of the crystal oscillator amplifier. The MCU internal system clock is derived

from the EXTAL input frequency. In full stop mode (PSTP = 0), the EXTAL pin is pulled down by an internal resistor of typical 200 kΩ.

**NOTE**

Freescale recommends an evaluation of the application board and chosen resonator or crystal by the resonator or crystal supplier.

Loop controlled circuit is not suited for overtone resonators and crystals.

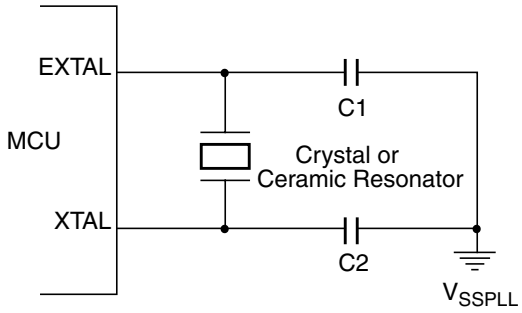
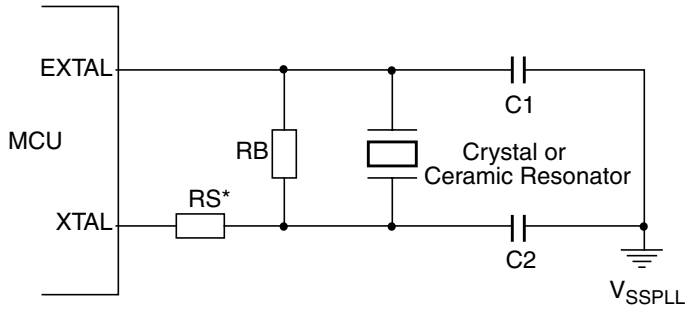



**Figure 12-2. Loop Controlled Pierce Oscillator Connections (LCP mode selected)**

**NOTE**

Full swing Pierce circuit is not suited for overtone resonators and crystals without a careful component selection.




* $R_S$ can be zero (shorted) when use with higher frequency crystals. Refer to manufacturer's data.

**Figure 12-3. Full Swing Pierce Oscillator Connections (FSP mode selected)**

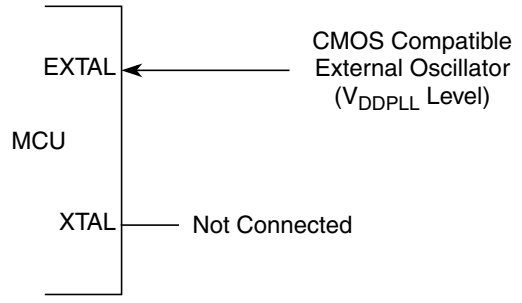



**Figure 12-4. External Clock Connections (FSP mode selected)**

## 12.3 Memory Map and Register Definition

The CRG contains the registers and associated bits for controlling and monitoring the oscillator module.

## 12.4 Functional Description

The XOSC module has control circuitry to maintain the crystal oscillator circuit voltage level to an optimal level which is determined by the amount of hysteresis being used and the maximum oscillation range.

The oscillator block has two external pins, EXTAL and XTAL. The oscillator input pin, EXTAL, is intended to be connected to either a crystal or an external clock source. The XTAL pin is an output signal that provides crystal circuit feedback.

A buffered EXTAL signal becomes the internal clock. To improve noise immunity, the oscillator is powered by the VDDPLL and VSSPLL power supply pins.

### 12.4.1 Gain Control

In LCP mode a closed loop control system will be utilized whereby the amplifier is modulated to keep the output waveform sinusoidal and to limit the oscillation amplitude. The output peak to peak voltage will be kept above twice the maximum hysteresis level of the input buffer. Electrical specification details are provided in the Electrical Characteristics appendix.

### 12.4.2 Clock Monitor

The clock monitor circuit is based on an internal RC time delay so that it can operate without any MCU clocks. If no OSCCLK edges are detected within this RC time delay, the clock monitor indicates failure which asserts self-clock mode or generates a system reset depending on the state of SCME bit. If the clock monitor is disabled or the presence of clocks is detected no failure is indicated.The clock monitor function is enabled/disabled by the CME control bit, described in the CRG block description chapter.

### 12.4.3 Wait Mode Operation

During wait mode, XOSC is not impacted.

### 12.4.4 Stop Mode Operation

XOSC is placed in a static state when the part is in stop mode except when pseudo-stop mode is enabled. During pseudo-stop mode, XOSC is not impacted.

# Chapter 13
# Analog-to-Digital Converter (ADC12B16CV1)

**Table 13-1. Revision History**

| Revision Number | Revision Date | Sections Affected | Description of Changes |
|---|---|---|---|
| V01.00 | 13 Oct. 2005 | | - Initial version |
| V01.01 | 04 Mar. 2008 | | corrected reference to DJM bit |

## 13.1 Introduction

The ADC12B16C is a 16-channel, 12-bit, multiplexed input successive approximation analog-to-digital converter. Refer to device electrical specifications for ATD accuracy.

### 13.1.1 Features

- 8-, 10-, or 12-bit resolution.
- Conversion in Stop Mode using internally generated clock
- Automatic return to low power after conversion sequence
- Automatic compare with interrupt for higher than or less/equal than programmable value
- Programmable sample time.
- Left/right justified result data.
- External trigger control.
- Sequence complete interrupt.
- Analog input multiplexer for 16 analog input channels.
- Special conversions for $V_{RH}$, $V_{RL}$, $(V_{RL}+V_{RH})/2$.
- 1-to-16 conversion sequence lengths.
- Continuous conversion mode.
- Multiple channel scans.
- Configurable external trigger functionality on any AD channel or any of four additional trigger inputs. The four additional trigger inputs can be chip external or internal. Refer to device specification for availability and connectivity.
- Configurable location for channel wrap around (when converting multiple channels in a sequence).

## 13.1.2 Modes of Operation

### 13.1.2.1 Conversion Modes

There is software programmable selection between performing **single** or **continuous conversion** on a **single channel** or **multiple channels**.

### 13.1.2.2 MCU Operating Modes

- **Stop Mode**
  - **ICLKSTP=0 (in ATDCTL2 register)**
    Entering Stop Mode aborts any conversion sequence in progress and if a sequence was aborted restarts it after exiting stop mode. This has the same effect/consequences as starting a conversion sequence with write to ATDCTL5. So after exiting from stop mode with a previously aborted sequence all flags are cleared etc.
  - **ICLKSTP=1 (in ATDCTL2 register)**
    A/D conversion sequence seamless continues in Stop Mode based on the internally generated clock ICLK as ATD clock. For conversions during transition from Run to Stop Mode or vice versa the result is not written to the results register, no CCF flag is set and no compare is done. When converting in Stop Mode (ICLKSTP=1) an ATD Stop Recovery time $t_{ATDSTPRCV}$ is required to switch back to bus clock based ATDCLK when leaving Stop Mode. Do not access ATD registers during this time.
- **Wait Mode**
  ADC12B16C behaves same in Run and Wait Mode. For reduced power consumption continuos conversions should be aborted before entering Wait mode.
- **Freeze Mode**
  In Freeze Mode the ADC12B16C will either continue or finish or stop converting according to the FRZ1 and FRZ0 bits. This is useful for debugging and emulation.

### 13.1.3 Block Diagram

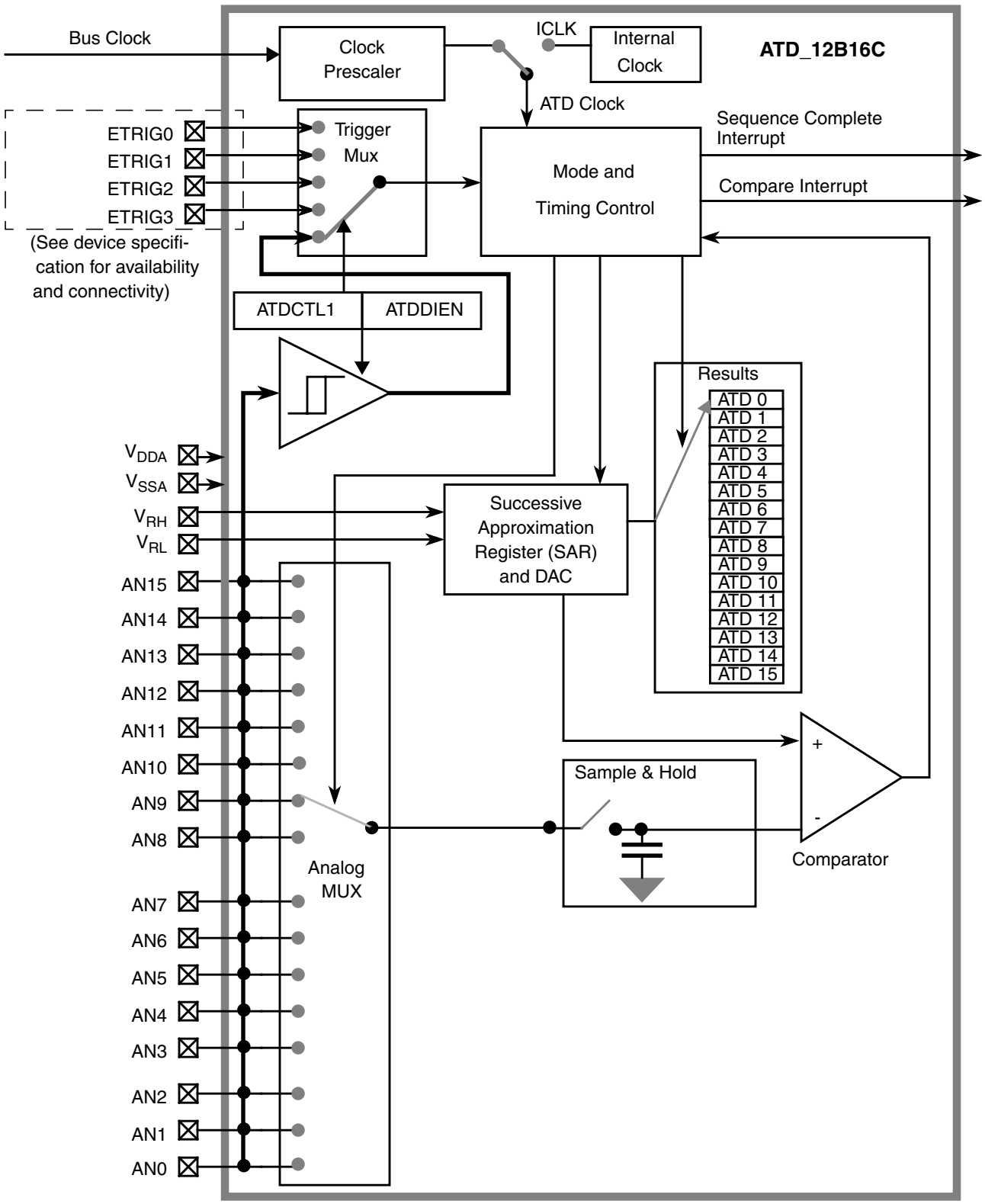



**Figure 13-1. ADC12B16C Block Diagram**

## 13.2 Signal Description

This section lists all inputs to the ADC12B16C block.

### 13.2.1 Detailed Signal Descriptions

#### 13.2.1.1 AN*x* (*x* = 15, 14, 13, 12, 11, 10, 9, 8, 7, 6, 5, 4, 3, 2, 1, 0)

This pin serves as the analog input Channel *x*. It can also be configured as digital port or external trigger for the ATD conversion.

#### 13.2.1.2 ETRIG3, ETRIG2, ETRIG1, ETRIG0

These inputs can be configured to serve as an external trigger for the ATD conversion.

Refer to device specification for availability and connection of these inputs!

#### 13.2.1.3 $V_{RH}$, $V_{RL}$

$V_{RH}$ is the high reference voltage, $V_{RL}$ is the low reference voltage for ATD conversion.

#### 13.2.1.4 $V_{DDA}$, $V_{SSA}$

These pins are the power supplies for the analog circuitry of the ADC12B16C block.

## 13.3 Memory Map and Register Definition

This section provides a detailed description of all registers accessible in the ADC12B16C.

### 13.3.1 Module Memory Map

Figure 13-2 gives an overview on all ADC12B16C registers.

**NOTE**

Register Address = Base Address + Address Offset, where the Base Address is defined at the MCU level and the Address Offset is defined at the module level.

| Address | Name | | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|---|---|
| 0x0000 | ATDCTL0 | R<br>W | Reserved | 0 | 0 | 0 | WRAP3 | WRAP2 | WRAP1 | WRAP0 |
| 0x0001 | ATDCTL1 | R<br>W | ETRIGSEL | SRES1 | SRES0 | SMP_DIS | ETRIGCH3 | ETRIGCH2 | ETRIGCH1 | ETRIGCH0 |
| 0x0002 | ATDCTL2 | R<br>W | 0 | AFFC | ICLKSTP | ETRIGLE | ETRIGP | ETRIGE | ASCIE | ACMPIE |

= Unimplemented or Reserved

**Figure 13-2. ADC12B16C Register Summary (Sheet 1 of 3)**

| Address | Name | | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|---|---|
| 0x0003 | ATDCTL3 | R<br>W | DJM | S8C | S4C | S2C | S1C | FIFO | FRZ1 | FRZ0 |
| 0x0004 | ATDCTL4 | R<br>W | SMP2 | SMP1 | SMP0 | PRS[4:0] | | | | |
| 0x0005 | ATDCTL5 | R<br>W | 0 | SC | SCAN | MULT | CD | CC | CB | CA |
| 0x0006 | ATDSTAT0 | R<br>W | SCF | 0 | ETORF | FIFOR | CC3 | CC2 | CC1 | CC0 |
| 0x0007 | Unimple-mented | R<br>W | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0x0008 | ATDCMPEH | R<br>W | CMPE[15:8] | | | | | | | |
| 0x0009 | ATDCMPEL | R<br>W | CMPE[7:0] | | | | | | | |
| 0x000A | ATDSTAT2H | R<br>W | CCF[15:8] | | | | | | | |
| 0x000B | ATDSTAT2L | R<br>W | CCF[7:0] | | | | | | | |
| 0x000C | ATDDIENH | R<br>W | IEN[15:8] | | | | | | | |
| 0x000D | ATDDIENL | R<br>W | IEN[7:0] | | | | | | | |
| 0x000E | ATDCMPHTH | R<br>W | CMPHT[15:8] | | | | | | | |
| 0x000F | ATDCMPHTL | R<br>W | CMPHT[7:0] | | | | | | | |
| 0x0010 | ATDDR0 | R<br>W | See Section 13.3.2.12.1, “Left Justified Result Data (DJM=0)” and Section 13.3.2.12.2, “Right Justified Result Data (DJM=1)” | | | | | | | |
| 0x0012 | ATDDR1 | R<br>W | See Section 13.3.2.12.1, “Left Justified Result Data (DJM=0)” and Section 13.3.2.12.2, “Right Justified Result Data (DJM=1)” | | | | | | | |
| 0x0014 | ATDDR2 | R<br>W | See Section 13.3.2.12.1, “Left Justified Result Data (DJM=0)” and Section 13.3.2.12.2, “Right Justified Result Data (DJM=1)” | | | | | | | |
| 0x0016 | ATDDR3 | R<br>W | See Section 13.3.2.12.1, “Left Justified Result Data (DJM=0)” and Section 13.3.2.12.2, “Right Justified Result Data (DJM=1)” | | | | | | | |
| 0x0018 | ATDDR4 | R<br>W | See Section 13.3.2.12.1, “Left Justified Result Data (DJM=0)” and Section 13.3.2.12.2, “Right Justified Result Data (DJM=1)” | | | | | | | |
| 0x001A | ATDDR5 | R<br>W | See Section 13.3.2.12.1, “Left Justified Result Data (DJM=0)” and Section 13.3.2.12.2, “Right Justified Result Data (DJM=1)” | | | | | | | |
| 0x001C | ATDDR6 | R<br>W | See Section 13.3.2.12.1, “Left Justified Result Data (DJM=0)” and Section 13.3.2.12.2, “Right Justified Result Data (DJM=1)” | | | | | | | |
| 0x001E | ATDDR7 | R<br>W | See Section 13.3.2.12.1, “Left Justified Result Data (DJM=0)” and Section 13.3.2.12.2, “Right Justified Result Data (DJM=1)” | | | | | | | |
| 0x0020 | ATDDR8 | R<br>W | See Section 13.3.2.12.1, “Left Justified Result Data (DJM=0)” and Section 13.3.2.12.2, “Right Justified Result Data (DJM=1)” | | | | | | | |
| 0x0022 | ATDDR9 | R<br>W | See Section 13.3.2.12.1, “Left Justified Result Data (DJM=0)” and Section 13.3.2.12.2, “Right Justified Result Data (DJM=1)” | | | | | | | |

= Unimplemented or Reserved

**Figure 13-2. ADC12B16C Register Summary (Sheet 2 of 3)**

| Address | Name | | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|---|---|
| 0x0024 | ATDDR10 | R<br>W | See Section 13.3.2.12.1, “Left Justified Result Data (DJM=0)” and Section 13.3.2.12.2, “Right Justified Result Data (DJM=1)” | | | | | | | |
| 0x0026 | ATDDR11 | R<br>W | See Section 13.3.2.12.1, “Left Justified Result Data (DJM=0)” and Section 13.3.2.12.2, “Right Justified Result Data (DJM=1)” | | | | | | | |
| 0x0028 | ATDDR12 | R<br>W | See Section 13.3.2.12.1, “Left Justified Result Data (DJM=0)” and Section 13.3.2.12.2, “Right Justified Result Data (DJM=1)” | | | | | | | |
| 0x002A | ATDDR13 | R<br>W | See Section 13.3.2.12.1, “Left Justified Result Data (DJM=0)” and Section 13.3.2.12.2, “Right Justified Result Data (DJM=1)” | | | | | | | |
| 0x002C | ATDDR14 | R<br>W | See Section 13.3.2.12.1, “Left Justified Result Data (DJM=0)” and Section 13.3.2.12.2, “Right Justified Result Data (DJM=1)” | | | | | | | |
| 0x002E | ATDDR15 | R<br>W | See Section 13.3.2.12.1, “Left Justified Result Data (DJM=0)” and Section 13.3.2.12.2, “Right Justified Result Data (DJM=1)” | | | | | | | |

= Unimplemented or Reserved

**Figure 13-2. ADC12B16C Register Summary (Sheet 3 of 3)**

## 13.3.2 Register Descriptions

This section describes in address order all the ADC12B16C registers and their individual bits.

### 13.3.2.1 ATD Control Register 0 (ATDCTL0)

Writes to this register will abort current conversion sequence.

Module Base + 0x0000

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | Reserved | 0 | 0 | 0 | WRAP3 | WRAP2 | WRAP1 | WRAP0 |
| W | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 |

= Unimplemented or Reserved

**Figure 13-3. ATD Control Register 0 (ATDCTL0)**

Read: Anytime

Write: Anytime, in special modes always write 0 to Reserved Bit 7.

**Table 13-2. ATDCTL0 Field Descriptions**

| Field | Description |
|---|---|
| | |
| 3-0<br>WRAP[3-0] | **Wrap Around Channel Select Bits** — These bits determine the channel for wrap around when doing multi-channel conversions. The coding is summarized in Table 13-3. |

**Table 13-3. Multi-Channel Wrap Around Coding**

| WRAP3 | WRAP2 | WRAP1 | WRAP0 | Multiple Channel Conversions (MULT = 1) Wraparound to AN0 after Converting |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | Reserved(1) |

**Table 13-3. Multi-Channel Wrap Around Coding**

| WRAP3 | WRAP2 | WRAP1 | WRAP0 | Multiple Channel Conversions (MULT = 1) Wraparound to AN0 after Converting |
|---|---|---|---|---|
| 0 | 0 | 0 | 1 | AN1 |
| 0 | 0 | 1 | 0 | AN2 |
| 0 | 0 | 1 | 1 | AN3 |
| 0 | 1 | 0 | 0 | AN4 |
| 0 | 1 | 0 | 1 | AN5 |
| 0 | 1 | 1 | 0 | AN6 |
| 0 | 1 | 1 | 1 | AN7 |
| 1 | 0 | 0 | 0 | AN8 |
| 1 | 0 | 0 | 1 | AN9 |
| 1 | 0 | 1 | 0 | AN10 |
| 1 | 0 | 1 | 1 | AN11 |
| 1 | 1 | 0 | 0 | AN12 |
| 1 | 1 | 0 | 1 | AN13 |
| 1 | 1 | 1 | 0 | AN14 |
| 1 | 1 | 1 | 1 | AN15 |

1. If only AN0 should be converted use MULT=0.

### 13.3.2.2 ATD Control Register 1 (ATDCTL1)

Writes to this register will abort current conversion sequence.

Module Base + 0x0001

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | ETRIGSEL | SRES1 | SRES0 | SMP_DIS | ETRIGCH3 | ETRIGCH2 | ETRIGCH1 | ETRIGCH0 |
| Reset | 0 | 0 | 1 | 0 | 1 | 1 | 1 | 1 |

**Figure 13-4. ATD Control Register 1 (ATDCTL1)**

Read: Anytime

Write: Anytime

**Table 13-4. ATDCTL1 Field Descriptions**

| Field | Description |
|---|---|
| 7<br>ETRIGSEL | **External Trigger Source Select** — This bit selects the external trigger source to be either one of the AD channels or one of the ETRIG3-0 inputs. See device specification for availability and connectivity of ETRIG3-0 inputs. If a particular ETRIG3-0 input option is not available, writing a 1 to ETRISEL only sets the bit but has not effect, this means that one of the AD channels (selected by ETRIGCH3-0) is configured as the source for external trigger. The coding is summarized in Table 13-6. |
| 6–5<br>SRES[1:0] | **A/D Resolution Select** — These bits select the resolution of A/D conversion results. See Table 13-5 for coding. |

**Table 13-4. ATDCTL1 Field Descriptions (continued)**

| Field | Description |
|---|---|
| 4<br>SMP_DIS | **Discharge Before Sampling Bit**<br>0 No discharge before sampling.<br>1 The internal sample capacitor is discharged before sampling the channel. This adds 2 ATD clock cycles to the sampling time. This can help to detect an open circuit instead of measuring the previous sampled channel. |
| 3–0<br>ETRIGCH[3:0] | **External Trigger Channel Select** — These bits select one of the AD channels or one of the ETRIG3-0 inputs as source for the external trigger. The coding is summarized in Table 13-6. |

**Table 13-5. A/D Resolution Coding**

| SRES1 | SRES0 | A/D Resolution |
|---|---|---|
| 0 | 0 | 8-bit data |
| 0 | 1 | 10-bit data |
| 1 | 0 | 12-bit data |
| 1 | 1 | Reserved |

**Table 13-6. External Trigger Channel Select Coding**

| ETRIGSEL | ETRIGCH3 | ETRIGCH2 | ETRIGCH1 | ETRIGCH0 | External trigger source is |
|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | AN0 |
| 0 | 0 | 0 | 0 | 1 | AN1 |
| 0 | 0 | 0 | 1 | 0 | AN2 |
| 0 | 0 | 0 | 1 | 1 | AN3 |
| 0 | 0 | 1 | 0 | 0 | AN4 |
| 0 | 0 | 1 | 0 | 1 | AN5 |
| 0 | 0 | 1 | 1 | 0 | AN6 |
| 0 | 0 | 1 | 1 | 1 | AN7 |
| 0 | 1 | 0 | 0 | 0 | AN8 |
| 0 | 1 | 0 | 0 | 1 | AN9 |
| 0 | 1 | 0 | 1 | 0 | AN10 |
| 0 | 1 | 0 | 1 | 1 | AN11 |
| 0 | 1 | 1 | 0 | 0 | AN12 |
| 0 | 1 | 1 | 0 | 1 | AN13 |
| 0 | 1 | 1 | 1 | 0 | AN14 |
| 0 | 1 | 1 | 1 | 1 | AN15 |
| 1 | 0 | 0 | 0 | 0 | ETRIG0[(1)] |
| 1 | 0 | 0 | 0 | 1 | ETRIG1[1] |
| 1 | 0 | 0 | 1 | 0 | ETRIG2[1] |
| 1 | 0 | 0 | 1 | 1 | ETRIG3[1] |
| 1 | 0 | 1 | X | X | Reserved |
| 1 | 1 | X | X | X | Reserved |

1. Only if ETRIG3-0 input option is available (see device specification), else ETRISEL is ignored, that means external trigger source is still on one of the AD channels selected by ETRIGCH3-0

### 13.3.2.3 ATD Control Register 2 (ATDCTL2)

Writes to this register will abort current conversion sequence.

Module Base + 0x0002

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | 0 | AFFC | ICLKSTP | ETRIGLE | ETRIGP | ETRIGE | ASCIE | ACMPIE |
| W | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented or Reserved

**Figure 13-5. ATD Control Register 2 (ATDCTL2)**

Read: Anytime

Write: Anytime

**Table 13-7. ATDCTL2 Field Descriptions**

| Field | Description |
|---|---|
| 6<br>AFFC | **ATD Fast Flag Clear All**<br>0 ATD flag clearing done by write 1 to respective CCF[*n*] flag.<br>1 Changes all ATD conversion complete flags to a fast clear sequence.<br>For compare disabled (CMPE[*n*]=0) a read access to the result register will cause the associated CCF[*n*] flag to clear automatically.<br>For compare enabled (CMPE[*n*]=1) a write access to the result register will cause the associated CCF[*n*] flag to clear automatically. |
| 5<br>ICLKSTP | **Internal Clock in Stop Mode Bit** — This bit enables A/D conversions in stop mode. When going into stop mode and ICLKSTP=1 the ATD conversion clock is automatically switched to the internally generated clock ICLK. Current conversion sequence will seamless continue. Conversion speed will change from prescaled bus frequency to the ICLK frequency (see ATD Electrical Characteristics in device description). The prescaler bits PRS4-0 in ATDCTL4 have no effect on the ICLK frequency. For conversions during stop mode the automatic compare interrupt or the sequence complete interrupt can be used to inform software handler about changing A/D values. External trigger will not work while converting in stop mode. For conversions during transition from Run to Stop Mode or vice versa the result is not written to the results register, no CCF flag is set and no compare is done. When converting in Stop Mode (ICLKSTP=1) an ATD Stop Recovery time $t_{ATDSTPRCV}$ is required to switch back to bus clock based ATDCLK when leaving Stop Mode. Do not access ATD registers during this time.<br>0 If A/D conversion sequence is ongoing when going into stop mode, the actual conversion sequence will be aborted and automatically restarted when exiting stop mode.<br>1 A/D continues to convert in stop mode using internally generated clock (ICLK) |
| 4<br>ETRIGLE | **External Trigger Level/Edge Control** — This bit controls the sensitivity of the external trigger signal. See Table 13-8 for details. |
| 3<br>ETRIGP | **External Trigger Polarity** — This bit controls the polarity of the external trigger signal. See Table 13-8 for details. |
| 2<br>ETRIGE | **External Trigger Mode Enable** — This bit enables the external trigger on one of the AD channels or one of the ETRIG3-0 inputs as described in Table 13-6. If external trigger source is one of the AD channels, the digital input buffer of this channel is enabled. The external trigger allows to synchronize the start of conversion with external events. External trigger will not work while converting in stop mode.<br>0 Disable external trigger<br>1 Enable external trigger |

**Table 13-7. ATDCTL2 Field Descriptions (continued)**

| Field | Description |
|---|---|
| 1<br>ASCIE | **ATD Sequence Complete Interrupt Enable**<br>0 ATD Sequence Complete interrupt requests are disabled.<br>1 ATD Sequence Complete interrupt will be requested whenever SCF=1 is set. |
| 0<br>ACMPIE | **ATD Compare Interrupt Enable** — If automatic compare is enabled for conversion *n* (CMPE[*n*]=1 in ATDCMPE register) this bit enables the compare interrupt. If the CCF[*n*] flag is set (showing a successful compare for conversion *n*), the compare interrupt is triggered.<br>0 ATD Compare interrupt requests are disabled.<br>1 For the conversions in a sequence for which automatic compare is enabled (CMPE[*n*]=1), ATD Compare Interrupt will be requested whenever any of the respective CCF flags is set. |

**Table 13-8. External Trigger Configurations**

| ETRIGLE | ETRIGP | External Trigger Sensitivity |
|---|---|---|
| 0 | 0 | Falling edge |
| 0 | 1 | Rising edge |
| 1 | 0 | Low level |
| 1 | 1 | High level |

### 13.3.2.4 ATD Control Register 3 (ATDCTL3)

Writes to this register will abort current conversion sequence.

Module Base + 0x0003

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | DJM | S8C | S4C | S2C | S1C | FIFO | FRZ1 | FRZ0 |
| Reset | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented or Reserved

**Figure 13-6. ATD Control Register 3 (ATDCTL3)**

Read: Anytime

Write: Anytime

| Field | Description |
|---|---|
| 7<br>DJM | **Result Register Data Justification** — Result data format is always unsigned. This bit controls justification of conversion data in the result registers.<br>0 Left justified data in the result registers.<br>1 Right justified data in the result registers.<br>Table 13-10 gives examples ATD results for an input signal range between 0 and 5.12 Volts. |

**Table 13-9. ATDCTL3 Field Descriptions**

| Field | Description |
|---|---|
| 6–3<br>S8C, S4C,<br>S2C, S1C | **Conversion Sequence Length** — These bits control the number of conversions per sequence. Table 13-11 shows all combinations. At reset, S4C is set to 1 (sequence length is 4). This is to maintain software continuity to HC12 family. |
| 2<br>FIFO | **Result Register FIFO Mode** — If this bit is zero (non-FIFO mode), the A/D conversion results map into the result registers based on the conversion sequence; the result of the first conversion appears in the first result register (ATDDR0), the second result in the second result register (ATDDR1), and so on.<br>If this bit is one (FIFO mode) the conversion counter is not reset at the beginning or ending of a conversion sequence; sequential conversion results are placed in consecutive result registers. In a continuously scanning conversion sequence, the result register counter will wrap around when it reaches the end of the result register file. The conversion counter value (CC3-0 in ATDSTAT0) can be used to determine where in the result register file, the current conversion result will be placed.<br>Aborting a conversion or starting a new conversion clears the conversion counter even if FIFO=1. So the first result of a new conversion sequence, started by writing to ATDCTL5, will always be place in the first result register (ATDDDR0). Intended usage of FIFO mode is continuos conversion (SCAN=1) or triggered conversion (ETRIG=1).<br>Which result registers hold valid data can be tracked using the conversion complete flags. Fast flag clear mode may or may not be useful in a particular application to track valid data.<br>If this bit is one, automatic compare of result registers is always disabled, that is ADC12B16C will behave as if ACMPIE and all CPME[*n*] were zero.<br>0 Conversion results are placed in the corresponding result register up to the selected sequence length.<br>1 Conversion results are placed in consecutive result registers (wrap around at end). |
| 1–0<br>FRZ[1:0] | **Background Debug Freeze Enable** — When debugging an application, it is useful in many cases to have the ATD pause when a breakpoint (Freeze Mode) is encountered. These 2 bits determine how the ATD will respond to a breakpoint as shown in Table 13-12. Leakage onto the storage node and comparator reference capacitors may compromise the accuracy of an immediately frozen conversion depending on the length of the freeze period. |

**Table 13-9. ATDCTL3 Field Descriptions (continued)**

**Table 13-10. Examples of ideal decimal ATD Results**

| Input Signal<br>$V_{RL}$ = 0 Volts<br>$V_{RH}$ = 5.12 Volts | 8-Bit<br>Codes<br>(resolution=20mV) | 10-Bit<br>Codes<br>(resolution=5mV) | 12-Bit<br>Codes<br>(transfer curve has<br>1.25mV offset)<br>(resolution=1.25mV) |
|---|---|---|---|
| 5.120 Volts | 255 | 1023 | 4095 |
| ... | ... | ... | ... |
| 0.022 | 1 | 4 | 17 |
| 0.020 | 1 | 4 | 16 |
| 0.018 | 1 | 4 | 14 |
| 0.016 | 1 | 3 | 12 |
| 0.014 | 1 | 3 | 11 |
| 0.012 | 1 | 2 | 9 |
| 0.010 | 1 | 2 | 8 |
| 0.008 | 0 | 2 | 6 |
| 0.006 | 0 | 1 | 4 |
| 0.004 | 0 | 1 | 3 |
| 0.003 | 0 | 0 | 2 |
| 0.002 | 0 | 0 | 1 |
| 0.000 | 0 | 0 | 0 |

**Table 13-11. Conversion Sequence Length Coding**

| S8C | S4C | S2C | S1C | Number of Conversions per Sequence |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 16 |
| 0 | 0 | 0 | 1 | 1 |
| 0 | 0 | 1 | 0 | 2 |
| 0 | 0 | 1 | 1 | 3 |
| 0 | 1 | 0 | 0 | 4 |
| 0 | 1 | 0 | 1 | 5 |
| 0 | 1 | 1 | 0 | 6 |
| 0 | 1 | 1 | 1 | 7 |
| 1 | 0 | 0 | 0 | 8 |
| 1 | 0 | 0 | 1 | 9 |
| 1 | 0 | 1 | 0 | 10 |
| 1 | 0 | 1 | 1 | 11 |
| 1 | 1 | 0 | 0 | 12 |
| 1 | 1 | 0 | 1 | 13 |
| 1 | 1 | 1 | 0 | 14 |
| 1 | 1 | 1 | 1 | 15 |

**Table 13-12. ATD Behavior in Freeze Mode (Breakpoint)**

| FRZ1 | FRZ0 | Behavior in Freeze Mode |
|---|---|---|
| 0 | 0 | Continue conversion |
| 0 | 1 | Reserved |
| 1 | 0 | Finish current conversion, then freeze |
| 1 | 1 | Freeze Immediately |

### 13.3.2.5 ATD Control Register 4 (ATDCTL4)

Writes to this register will abort current conversion sequence.

Module Base + 0x0004

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | SMP2 | SMP1 | SMP0 | PRS[4:0] | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 |

**Figure 13-7. ATD Control Register 4 (ATDCTL4)**

Read: Anytime

Write: Anytime

**Table 13-13. ATDCTL4 Field Descriptions**

| Field | Description |
|---|---|
| 7–5<br>SMP[2:0] | **Sample Time Select** — These three bits select the length of the sample time in units of ATD conversion clock cycles. Note that the ATD conversion clock period is itself a function of the prescaler value (bits PRS4-0). Table 13-14 lists the available sample time lengths. |
| 4–0<br>PRS[4:0] | **ATD Clock Prescaler** — These 5 bits are the binary prescaler value PRS. The ATD conversion clock frequency is calculated as follows:<br>$f_{ATDCLK} = \frac{f_{BUS}}{2 \times (PRS + 1)}$<br>Refer to Device Specification for allowed frequency range of $f_{ATDCLK}$. |

**Table 13-14. Sample Time Select**

| SMP2 | SMP1 | SMP0 | Sample Time in Number of ATD Clock Cycles |
|---|---|---|---|
| 0 | 0 | 0 | 4 |
| 0 | 0 | 1 | 6 |
| 0 | 1 | 0 | 8 |
| 0 | 1 | 1 | 10 |
| 1 | 0 | 0 | 12 |
| 1 | 0 | 1 | 16 |
| 1 | 1 | 0 | 20 |
| 1 | 1 | 1 | 24 |

### 13.3.2.6 ATD Control Register 5 (ATDCTL5)

Writes to this register will abort current conversion sequence and start a new conversion sequence. If external trigger is enabled (ETRIGE=1) an initial write to ATDCTL5 is required to allow starting of a conversion sequence which will then occur on each trigger event. Start of conversion means the beginning of the sampling phase.

Module Base + 0x0005

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | 0 | SC | SCAN | MULT | CD | CC | CB | CA |
| W | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 13-8. ATD Control Register 5 (ATDCTL5)**

Read: Anytime

Write: Anytime

**Table 13-15. ATDCTL5 Field Descriptions**

| Field | Description |
|---|---|
| 6<br>SC | **Special Channel Conversion Bit** — If this bit is set, then special channel conversion can be selected using CD, CC, CB and CA of ATDCTL5. Table 13-16 lists the coding.<br>0 Special channel conversions disabled<br>1 Special channel conversions enabled |
| 5<br>SCAN | **Continuous Conversion Sequence Mode** — This bit selects whether conversion sequences are performed continuously or only once. If external trigger is enabled (ETRIGE=1) setting this bit has no effect, that means external trigger always starts a single conversion sequence.<br>0 Single conversion sequence<br>1 Continuous conversion sequences (scan mode) |
| 4<br>MULT | **Multi-Channel Sample Mode** — When MULT is 0, the ATD sequence controller samples only from the specified analog input channel for an entire conversion sequence. The analog channel is selected by channel selection code (control bits CD/CC/CB/CA located in ATDCTL5). When MULT is 1, the ATD sequence controller samples across channels. The number of channels sampled is determined by the sequence length value (S8C, S4C, S2C, S1C). The first analog channel examined is determined by channel selection code (CD, CC, CB, CA control bits); subsequent channels sampled in the sequence are determined by incrementing the channel selection code or wrapping around to AN0 (channel 0).<br>0 Sample only one channel<br>1 Sample across several channels |
| 3–0<br>CD, CC,<br>CB, CA | **Analog Input Channel Select Code** — These bits select the analog input channel(s) whose signals are sampled and converted to digital codes. Table 13-16 lists the coding used to select the various analog input channels.<br>In the case of single channel conversions (MULT=0), this selection code specifies the channel to be examined.<br>In the case of multiple channel conversions (MULT=1), this selection code specifies the first channel to be examined in the conversion sequence. Subsequent channels are determined by incrementing the channel selection code or wrapping around to AN0 (after converting the channel defined by the Wrap Around Channel Select Bits WRAP3-0 in ATDCTL0). In case of starting with a channel number higher than the one defined by WRAP3-0 the first wrap around will be AN15 to AN0. |

**Table 13-16. Analog Input Channel Select Coding**

| SC | CD | CC | CB | CA | Analog Input Channel |
|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | AN0 |
| | 0 | 0 | 0 | 1 | AN1 |
| | 0 | 0 | 1 | 0 | AN2 |
| | 0 | 0 | 1 | 1 | AN3 |
| | 0 | 1 | 0 | 0 | AN4 |
| | 0 | 1 | 0 | 1 | AN5 |
| | 0 | 1 | 1 | 0 | AN6 |
| | 0 | 1 | 1 | 1 | AN7 |
| | 1 | 0 | 0 | 0 | AN8 |
| | 1 | 0 | 0 | 1 | AN9 |
| | 1 | 0 | 1 | 0 | AN10 |
| | 1 | 0 | 1 | 1 | AN11 |
| | 1 | 1 | 0 | 0 | AN12 |
| | 1 | 1 | 0 | 1 | AN13 |
| | 1 | 1 | 1 | 0 | AN14 |
| | 1 | 1 | 1 | 1 | AN15 |
| 1 | 0 | 0 | 0 | 0 | Reserved |
| | 0 | 0 | 0 | 1 | Reserved |
| | 0 | 0 | 1 | X | Reserved |
| | 0 | 1 | 0 | 0 | $V_{RH}$ |
| | 0 | 1 | 0 | 1 | $V_{RL}$ |
| | 0 | 1 | 1 | 0 | $(V_{RH}+V_{RL}) / 2$ |
| | 0 | 1 | 1 | 1 | Reserved |
| | 1 | X | X | X | Reserved |

### 13.3.2.7 ATD Status Register 0 (ATDSTAT0)

This register contains the Sequence Complete Flag, overrun flags for external trigger and FIFO mode, and the conversion counter.

Module Base + 0x0006

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | SCF | 0 | ETORF | FIFOR | CC3 | CC2 | CC1 | CC0 |
| W | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented or Reserved

**Figure 13-9. ATD Status Register 0 (ATDSTAT0)**

Read: Anytime

Write: Anytime (No effect on (CC3, CC2, CC1, CC0))

**Table 13-17. ATDSTAT0 Field Descriptions**

| Field | Description |
|---|---|
| 7<br>SCF | **Sequence Complete Flag** — This flag is set upon completion of a conversion sequence. If conversion sequences are continuously performed (SCAN=1), the flag is set after each one is completed. This flag is cleared when one of the following occurs:<br>A) Write “1” to SCF<br>B) Write to ATDCTL5 (a new conversion sequence is started)<br>C) If AFFC=1 and read of a result register<br>0 Conversion sequence not completed<br>1 Conversion sequence has completed |
| 5<br>ETORF | **External Trigger Overrun Flag** — While in edge trigger mode (ETRIGLE=0), if additional active edges are detected while a conversion sequence is in process the overrun flag is set. This flag is cleared when one of the following occurs:<br>A) Write “1” to ETORF<br>B) Write to ATDCTL0,1,2,3,4, ATDCMPE or ATDCMPHT (a conversion sequence is aborted)<br>C) Write to ATDCTL5 (a new conversion sequence is started)<br>0 No External trigger over run error has occurred<br>1 External trigger over run error has occurred |
| 4<br>FIFOR | **Result Register Over Run Flag** — This bit indicates that a result register has been written to before its associated conversion complete flag (CCF) has been cleared. This flag is most useful when using the FIFO mode because the flag potentially indicates that result registers are out of sync with the input channels. However, it is also practical for non-FIFO modes, and indicates that a result register has been over written before it has been read (i.e. the old data has been lost). This flag is cleared when one of the following occurs:<br>A) Write “1” to FIFOR<br>B) Write to ATDCTL0,1,2,3,4, ATDCMPE or ATDCMPHT (a conversion sequence is aborted)<br>C) Write to ATDCTL5 (a new conversion sequence is started)<br>0 No over run has occurred<br>1 Overrun condition exists (result register has been written while associated CCFx flag was still set) |
| 3–0<br>CC[3:0] | **Conversion Counter** — These 4 read-only bits are the binary value of the conversion counter. The conversion counter points to the result register that will receive the result of the current conversion. E.g. CC3=0, CC2=1, CC1=1, CC0=0 indicates that the result of the current conversion will be in ATD Result Register 6. If in non-FIFO mode (FIFO=0) the conversion counter is initialized to zero at the begin and end of the conversion sequence. If in FIFO mode (FIFO=1) the register counter is not initialized. The conversion counters wraps around when its maximum value is reached.<br>Aborting a conversion or starting a new conversion clears the conversion counter even if FIFO=1. |

### 13.3.2.8 ATD Compare Enable Register (ATDCMPE)

Writes to this register will abort current conversion sequence.

Read: Anytime

Write: Anytime

Module Base + 0x0008

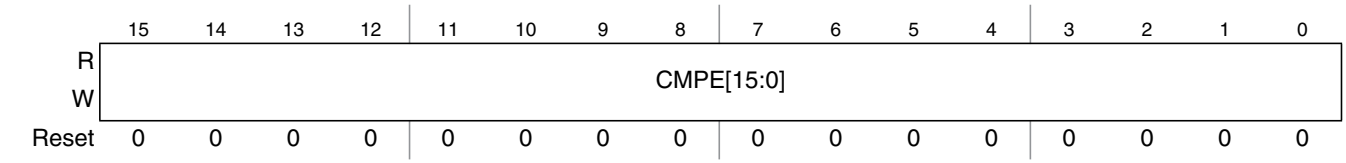



**Figure 13-10. ATD Compare Enable Register (ATDCMPE)**

**Table 13-18. ATDCMPE Field Descriptions**

| Field | Description |
|---|---|
| 15–0<br>CMPE[15:0] | **Compare Enable for Conversion Number *n* (*n*= 15, 14, 13, 12, 11, 10, 9, 8, 7, 6, 5, 4, 3, 2, 1, 0) of a Sequence** — These bits enable automatic compare of conversion results individually for conversions of a sequence. The sense of each comparison is determined by the CMPHT[*n*] bit in the ATDCMPHT register.<br>For each conversion number with CMPE[*n*]=1 do the following:<br>1) Write compare value to ATDDR*n* result register<br>2) Write compare operator with CMPHT[*n*] in ATDCPMHT register<br>CCF[*n*] in ATDSTAT2 register will flag individual success of any comparison.<br>0 No automatic compare<br>1 Automatic compare of results for conversion *n* of a sequence is enabled. |

### 13.3.2.9 ATD Status Register 2 (ATDSTAT2)

This read-only register contains the Conversion Complete Flags CCF[15:0].

Module Base + 0x000A

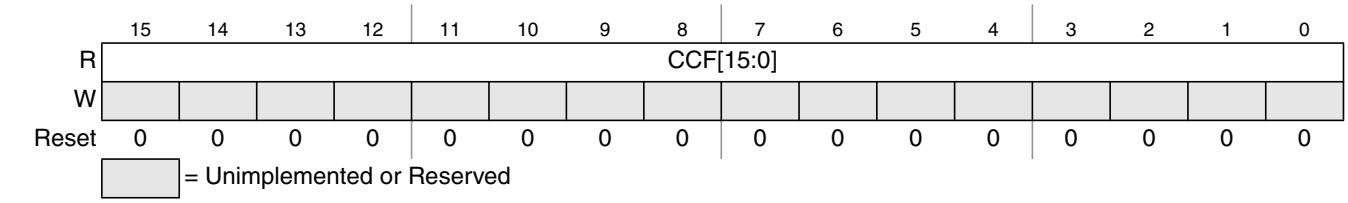



**Figure 13-11. ATD Status Register 2 (ATDSTAT2)**

Read: Anytime

Write: Anytime, no effect

**Table 13-19. ATDSTAT2 Field Descriptions**

| Field | Description |
|---|---|
| 15–0<br>CCF[15:0] | **Conversion Complete Flag *n* (*n*= 15, 14, 13, 12, 11, 10, 9, 8, 7, 6, 5, 4, 3, 2, 1, 0)** — A conversion complete flag is set at the end of each conversion in a sequence. The flags are associated with the conversion position in a sequence (and also the result register number). Therefore in non-fifo mode, CCF[8] is set when the ninth conversion in a sequence is complete and the result is available in result register ATDDR8; CCF[9] is set when the tenth conversion in a sequence is complete and the result is available in ATDDR9, and so forth.<br>If automatic compare of conversion results is enabled (CMPE[*n*]=1 in ATDCMPE), the conversion complete flag is only set if comparison with ATDDR*n* is true and if ACMPIE=1 a compare interrupt will be requested. In this case, as the ATDDR*n* result register is used to hold the compare value, the result will not be stored there at the end of the conversion but is lost.<br>A flag CCF[*n*] is cleared when one of the following occurs:<br>A) Write to ATDCTL5 (a new conversion sequence is started)<br>B) If AFFC=0, write “1” to CCF[*n*]<br>C) If AFFC=1 and CMPE[*n*]=0, read of result register ATDDR*n*<br>D) If AFFC=1 and CMPE[*n*]=1, write to result register ATDDR*n*<br>In case of a concurrent set and clear on CCF[*n*]: The clearing by method A) will overwrite the set. The clearing by methods B) or C) or D) will be overwritten by the set.<br>0 Conversion number *n* not completed or successfully compared<br>1 If (CMPE[*n*]=0): Conversion number *n* has completed. Result is ready in ATDDR*n*.<br>If (CMPE[*n*]=1): Compare for conversion result number *n* with compare value in ATDDR*n*, using compare operator CMPGT[*n*] is true. (No result available in ATDDR*n*) |

### 13.3.2.10 ATD Input Enable Register (ATDDIEN)

Module Base + 0x000C

| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | IEN[15:0] | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 13-12. ATD Input Enable Register (ATDDIEN)**

Read: Anytime

Write: Anytime

**Table 13-20. ATDDIEN Field Descriptions**

| Field | Description |
|---|---|
| 15–0<br>IEN[15:0] | **ATD Digital Input Enable on channel *x* (*x*= 15, 14, 13, 12, 11, 10, 9, 8, 7, 6, 5, 4, 3, 2, 1, 0)** — This bit controls the digital input buffer from the analog input pin (AN*x*) to the digital data register.<br>0 Disable digital input buffer to AN*x* pin<br>1 Enable digital input buffer on AN*x* pin.<br>**Note:** Setting this bit will enable the corresponding digital input buffer continuously. If this bit is set while simultaneously using it as an analog port, there is potentially increased power consumption because the digital input buffer maybe in the linear region. |

### 13.3.2.11 ATD Compare Higher Than Register (ATDCMPHT)

Writes to this register will abort current conversion sequence.

Read: Anytime

Write: Anytime

Module Base + 0x000E

| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | CMPHT[15:0] | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 13-13. ATD Compare Higher Than Register (ATDCMPHT)**

**Table 13-21. ATDCMPHT Field Descriptions**

| Field | Description |
|---|---|
| 15–0<br>CMPHT[15:0] | **Compare Operation Higher Than Enable for conversion number *n* (*n*= 15, 14, 13, 12, 11, 10, 9, 8, 7, 6, 5, 4, 3, 2, 1, 0) of a Sequence** — This bit selects the operator for comparison of conversion results.<br>0 If result of conversion *n* is **lower or same than** compare value in ATDDR*n*, this is flagged in ATDSTAT2<br>1 If result of conversion *n* is **higher than** compare value in ATDDR*n*, this is flagged in ATDSTAT2 |

### 13.3.2.12 ATD Conversion Result Registers (ATDDR*n*)

The A/D conversion results are stored in 16 result registers. Results are always in unsigned data representation. Left and right justification is selected using the DJM control bit in ATDCTL3.

If automatic compare of conversions results is enabled (CMPE[*n*]=1 in ATDCMPE), these registers must be written with the compare values in left or right justified format depending on the actual value of the DJM bit. In this case, as the ATDDR*n* register is used to hold the compare value, the result will not be stored there at the end of the conversion but is lost.

Read: Anytime

Write: Anytime

**NOTE**

For conversions not using automatic compare, results are stored in the result registers after each conversion. In this case avoid writing to ATDDRn except for initial values, because an A/D result might be overwritten.

#### 13.3.2.12.1 Left Justified Result Data (DJM=0)

Module Base +
0x0010 = ATDDR0, 0x0012 = ATDDR1, 0x0014 = ATDDR2, 0x0016 = ATDDR3
0x0018 = ATDDR4, 0x001A = ATDDR5, 0x001C = ATDDR6, 0x001E = ATDDR7
0x0020 = ATDDR8, 0x0022 = ATDDR9, 0x0024 = ATDDR10, 0x0026 = ATDDR11
0x0028 = ATDDR12, 0x002A = ATDDR13, 0x002C = ATDDR14, 0x002E = ATDDR15

| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | Bit 11 | Bit 10 | Bit 9 | Bit 8 | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | 0 | 0 | 0 | 0 |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 13-14. Left justified ATD conversion result register (ATDDR*n*)**

#### 13.3.2.12.2 Right Justified Result Data (DJM=1)

Module Base +
0x0010 = ATDDR0, 0x0012 = ATDDR1, 0x0014 = ATDDR2, 0x0016 = ATDDR3
0x0018 = ATDDR4, 0x001A = ATDDR5, 0x001C = ATDDR6, 0x001E = ATDDR7
0x0020 = ATDDR8, 0x0022 = ATDDR9, 0x0024 = ATDDR10, 0x0026 = ATDDR11
0x0028 = ATDDR12, 0x002A = ATDDR13, 0x002C = ATDDR14, 0x002E = ATDDR15

| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | 0 | 0 | Bit 11 | Bit 10 | Bit 9 | Bit 8 | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bi1 1 | Bit 0 |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 13-15. Right justified ATD conversion result register (ATDDR*n*)**

Table 13-16 shows how depending on the A/D resolution the conversion result is transferred to the ATD result registers. Compare is always done using all 12 bits of both the conversion result and the compare value in ATDDRn.

**Table 13-22. Conversion result mapping to ATDDRn**

| A/D resolution | DJM | conversion result mapping to ATDDR*n* |
|---|---|---|
| 8-bit data | 0 | Bit[11:4] = result, Bit[3:0]=0000 |
| 8-bit data | 1 | Bit[7:0] = result, Bit[11:8]=0000 |
| 10-bit data | 0 | Bit[11:2] = result, Bit[1:0]=00 |
| 10-bit data | 1 | Bit[9:0] = result, Bit[11:10]=00 |
| 12-bit data | X | Bit[11:0] = result |

## 13.4 Functional Description

The ADC12B16C is structured into an analog sub-block and a digital sub-block.

### 13.4.1 Analog Sub-Block

The analog sub-block contains all analog electronics required to perform a single conversion. Separate power supplies $V_{DDA}$ and $V_{SSA}$ allow to isolate noise of other MCU circuitry from the analog sub-block.

#### 13.4.1.1 Sample and Hold Machine

The Sample and Hold (S/H) Machine accepts analog signals from the external world and stores them as capacitor charge on a storage node.

During the sample process the analog input connects directly to the storage node.

The input analog signals are unipolar and must fall within the potential range of $V_{SSA}$ to $V_{DDA}$.

During the hold process the analog input is disconnected from the storage node.

#### 13.4.1.2 Analog Input Multiplexer

The analog input multiplexer connects one of the 16 external analog input channels to the sample and hold machine.

#### 13.4.1.3 Analog-to-Digital (A/D) Machine

The A/D Machine performs analog to digital conversions. The resolution is program selectable at either 8 or 10 or 12 bits. The A/D machine uses a successive approximation architecture. It functions by comparing the stored analog sample potential with a series of digitally generated analog potentials. By following a binary search algorithm, the A/D machine locates the approximating potential that is nearest to the sampled potential.

When not converting the A/D machine is automatically powered down.

Only analog input signals within the potential range of $V_{RL}$ to $V_{RH}$ (A/D reference potentials) will result in a non-railed digital output code.

## 13.4.2 Digital Sub-Block

This subsection explains some of the digital features in more detail. See Section 13.3.2, “Register Descriptions” for all details.

### 13.4.2.1 External Trigger Input

The external trigger feature allows the user to synchronize ATD conversions to the external environment events rather than relying on software to signal the ATD module when ATD conversions are to take place. The external trigger signal (out of reset ATD channel 15, configurable in ATDCTL1) is programmable to be edge or level sensitive with polarity control. Table 13-23 gives a brief description of the different combinations of control bits and their effect on the external trigger function.

**Table 13-23. External Trigger Control Bits**

| ETRIGLE | ETRIGP | ETRIGE | SCAN | Description |
|---|---|---|---|---|
| X | X | 0 | 0 | Ignores external trigger. Performs one conversion sequence and stops. |
| X | X | 0 | 1 | Ignores external trigger. Performs continuous conversion sequences. |
| 0 | 0 | 1 | X | Falling edge triggered. Performs one conversion sequence per trigger. |
| 0 | 1 | 1 | X | Rising edge triggered. Performs one conversion sequence per trigger. |
| 1 | 0 | 1 | X | Trigger active low. Performs continuous conversions while trigger is active. |
| 1 | 1 | 1 | X | Trigger active high. Performs continuous conversions while trigger is active. |

During a conversion, if additional active edges are detected the overrun error flag ETORF is set.

In either level or edge triggered modes, the first conversion begins when the trigger is received.

Once ETRIGE is enabled, conversions cannot be started by a write to ATDCTL5, but rather must be triggered externally.

If the level mode is active and the external trigger both de-asserts and re-asserts itself during a conversion sequence, this does not constitute an overrun. Therefore, the flag is not set. If the trigger is left asserted in level mode while a sequence is completing, another sequence will be triggered immediately.

#### 13.4.2.2 General-Purpose Digital Port Operation

The input channel pins can be multiplexed between analog and digital data. As analog inputs, they are multiplexed and sampled as analog channels to the A/D converter. The analog/digital multiplex operation is performed in the input pads. The input pad is always connected to the analog input channels of the ADC12B16C. The input pad signal is buffered to the digital port registers. This buffer can be turned on or off with the ATDDIEN register. This is important so that the buffer does not draw excess current when analog potentials are presented at its input.

## 13.5 Resets

At reset the ADC12B16C is in a power down state. The reset state of each individual bit is listed within the Register Description section (see Section 13.3.2, “Register Descriptions”) which details the registers and their bit-field.

## 13.6 Interrupts

The interrupts requested by the ADC12B16C are listed in Table 13-24. Refer to MCU specification for related vector address and priority.

**Table 13-24. ATD Interrupt Vectors**

| Interrupt Source | CCR Mask | Local Enable |
|---|---|---|
| Sequence Complete Interrupt | I bit | ASCIE in ATDCTL2 |
| Compare Interrupt | I bit | ACMPIE in ATDCTL2 |

See Section 13.3.2, “Register Descriptions” for further details.

# Chapter 14
# Enhanced Capture Timer (ECT16B8CV3)

**Table 14-1. Revision History**

| Revision Number | Revision Date | Sections Affected | Description of Changes |
|---|---|---|---|
| V03.06 | 05 Aug 2009 | 14.3.2.15/14-549<br>14.3.2.16/14-551<br>14.3.2.24/14-557<br>14.3.2.29/14-562<br>14.4.1.1.2/14-573 | update register PACTL bit4 PEDGE PT7 to IC7<br>update register PAFLG bit0 PAIF PT7 to IC7,update bit1 PAOVF PT3 to IC3<br>update register ICSYS bit3 TFMOD PTx to ICx<br>update register PBFLG bit1 PBOVF PT1 to IC1<br>update IC Queue Mode description. |
| V03.07 | 26 Aug 2009 | 14.3.2.2/14-536<br>14.3.2.3/14-536<br>14.3.2.4/14-537 | - Add description, ?a counter overflow when TTOV[7] is set?, to be the condition of channel 7 override event.<br>- Phrase the description of OC7M to make it more explicit |
| V03.08 | 04 May 2010 | 14.3.2.8/14-540<br>14.3.2.11/14-543 | - Add Table 14-11<br>- TCRE description, add Note and Figure 14-17 |

## 14.1 Introduction

The HCS12 enhanced capture timer module has the features of the HCS12 standard timer module enhanced by additional features in order to enlarge the field of applications, in particular for automotive ABS applications.

This design specification describes the standard timer as well as the additional features.

The basic timer consists of a 16-bit, software-programmable counter driven by a prescaler. This timer can be used for many purposes, including input waveform measurements while simultaneously generating an output waveform. Pulse widths can vary from microseconds to many seconds.

A full access for the counter registers or the input capture/output compare registers will take place in one clock cycle. Accessing high byte and low byte separately for all of these registers will not yield the same result as accessing them in one word.

### 14.1.1 Features

- 16-bit buffer register for four input capture (IC) channels.

- Four 8-bit pulse accumulators with 8-bit buffer registers associated with the four buffered IC channels. Configurable also as two 16-bit pulse accumulators.
- 16-bit modulus down-counter with 8-bit prescaler.
- Four user-selectable delay counters for input noise immunity increase.

### 14.1.2 Modes of Operation

- Stop — Timer and modulus counter are off since clocks are stopped.
- Freeze — Timer and modulus counter keep on running, unless the TSFRZ bit in the TSCR1 register is set to one.
- Wait — Counters keep on running, unless the TSWAI bit in the TSCR1 register is set to one.
- Normal — Timer and modulus counter keep on running, unless the TEN bit in the TSCR1 register or the MCEN bit in the MCCTL register are cleared.

### 14.1.3 Block Diagram

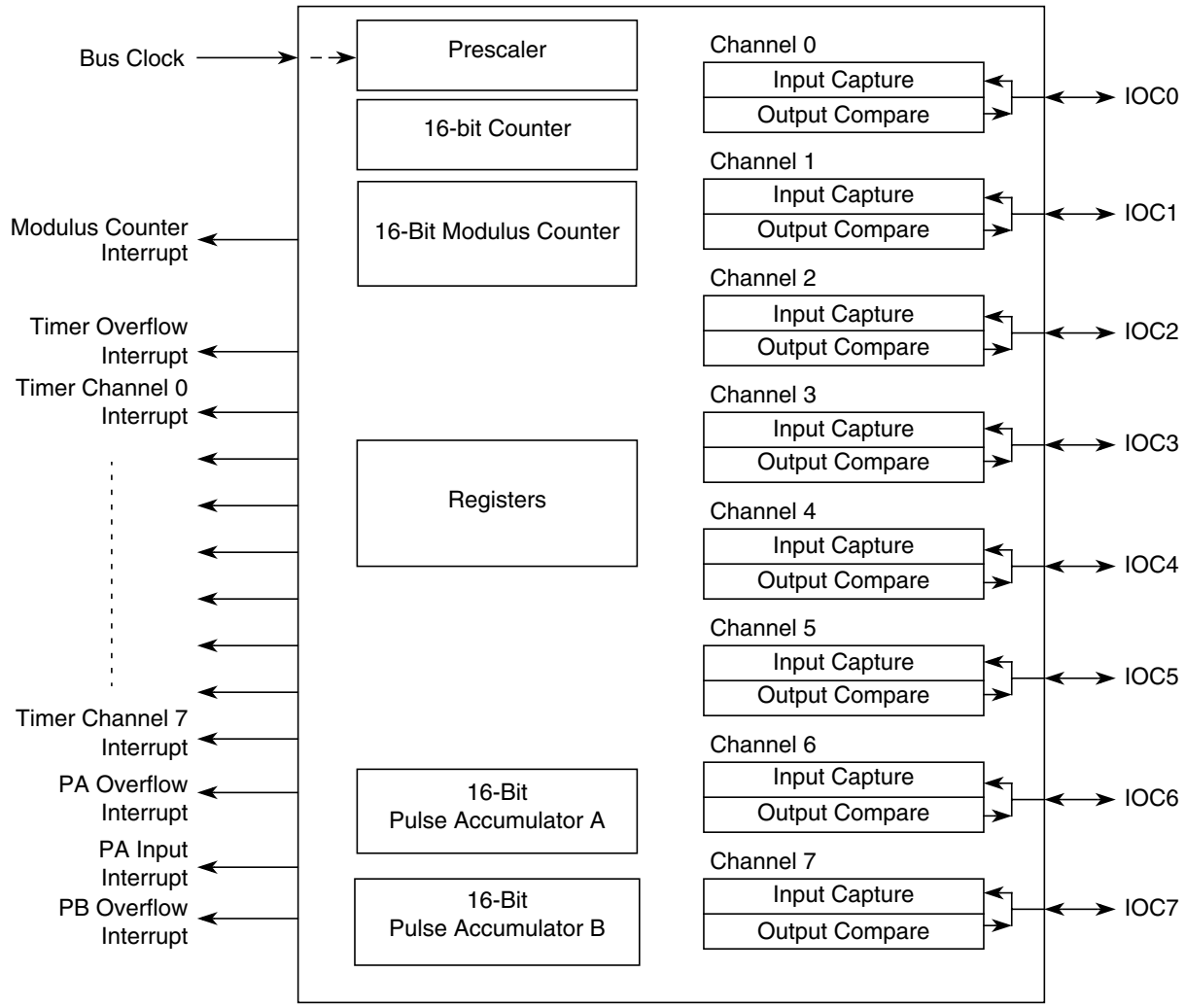



**Figure 14-1. ECT Block Diagram**

## 14.2 External Signal Description

The ECT module has a total of eight external pins.

### 14.2.1 IOC7 — Input Capture and Output Compare Channel 7

This pin serves as input capture or output compare for channel 7.

### 14.2.2 IOC6 — Input Capture and Output Compare Channel 6

This pin serves as input capture or output compare for channel 6.

### 14.2.3 IOC5 — Input Capture and Output Compare Channel 5

This pin serves as input capture or output compare for channel 5.

### 14.2.4 IOC4 — Input Capture and Output Compare Channel 4

This pin serves as input capture or output compare for channel 4.

### 14.2.5 IOC3 — Input Capture and Output Compare Channel 3

This pin serves as input capture or output compare for channel 3.

### 14.2.6 IOC2 — Input Capture and Output Compare Channel 2

This pin serves as input capture or output compare for channel 2.

### 14.2.7 IOC1 — Input Capture and Output Compare Channel 1

This pin serves as input capture or output compare for channel 1.

### 14.2.8 IOC0 — Input Capture and Output Compare Channel 0

This pin serves as input capture or output compare for channel 0.

**NOTE**

For the description of interrupts see Section 14.4.3, “Interrupts”.

## 14.3 Memory Map and Register Definition

This section provides a detailed description of all memory and registers.

### 14.3.1 Module Memory Map

The memory map for the ECT module is given below in the Table 14-2. The address listed for each register is the address offset. The total address for each register is the sum of the base address for the ECT module and the address offset for each register.

### 14.3.2 Register Descriptions

This section consists of register descriptions in address order. Each description includes a standard register diagram with an associated figure number. Details of register bit and field function follow the register diagrams, in bit order.

| Register Name | | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|---|
| 0x0000 TIOS | R | IOS7 | IOS6 | IOS5 | IOS4 | IOS3 | IOS2 | IOS1 | IOS0 |
| | W | | | | | | | | |
| 0x0001 CFORC | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | W | FOC7 | FOC6 | FOC5 | FOC4 | FOC3 | FOC2 | FOC1 | FOC0 |
| 0x0002 OC7M | R | OC7M7 | OC7M6 | OC7M5 | OC7M4 | OC7M3 | OC7M2 | OC7M1 | OC7M0 |
| | W | | | | | | | | |
| 0x0003 OC7D | R | OC7D7 | OC7D6 | OC7D5 | OC7D4 | OC7D3 | OC7D2 | OC7D1 | OC7D0 |
| | W | | | | | | | | |
| 0x0004 TCNT (High) | R | TCNT15 | TCNT14 | TCNT13 | TCNT12 | TCNT11 | TCNT10 | TCNT9 | TCNT8 |
| | W | | | | | | | | |
| 0x0005 TCNT (Low) | R | TCNT7 | TCNT6 | TCNT5 | TCNT4 | TCNT3 | TCNT2 | TCNT1 | TCNT0 |
| | W | | | | | | | | |
| 0x0006 TSCR1 | R | TEN | TSWAI | TSFRZ | TFFCA | PRNT | 0 | 0 | 0 |
| | W | | | | | | | | |
| 0x0007 TTOF | R | TOV7 | TOV6 | TOV5 | TOV4 | TOV3 | TOV2 | TOV1 | TOV0 |
| | W | | | | | | | | |
| 0x0008 TCTL1 | R | OM7 | OL7 | OM6 | OL6 | OM5 | OL5 | OM4 | OL4 |
| | W | | | | | | | | |
| 0x0009 TCTL2 | R | OM3 | OL3 | OM2 | OL2 | OM1 | OL1 | OM0 | OL0 |
| | W | | | | | | | | |
| 0x000A TCTL3 | R | EDG7B | EDG7A | EDG6B | EDG6A | EDG5B | EDG5A | EDG4B | EDG4A |
| | W | | | | | | | | |
| 0x000B TCTL4 | R | EDG3B | EDG3A | EDG2B | EDG2A | EDG1B | EDG1A | EDG0B | EDG0A |
| | W | | | | | | | | |
| 0x000C TIE | R | C7I | C6I | C5I | C4I | C3I | C2I | C1I | C0I |
| | W | | | | | | | | |

= Unimplemented or Reserved

**Figure 14-2. ECT Register Summary (Sheet 1 of 5)**

| Register Name | | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|---|
| 0x000D TSCR2 | R | TOI | 0 | 0 | 0 | TCRE | PR2 | PR1 | PR0 |
| | W | | | | | | | | |
| 0x000E TFLG1 | R | C7F | C6F | C5F | C4F | C3F | C2F | C1F | C0F |
| | W | | | | | | | | |
| 0x000F TFLG2 | R | TOF | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | W | | | | | | | | |
| 0x0010 TC0 (High) | R | Bit 15 | Bit 14 | Bit 13 | Bit 12 | Bit 11 | Bit 10 | Bit 9 | Bit 8 |
| | W | | | | | | | | |
| 0x0011 TC0 (Low) | R | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
| | W | | | | | | | | |
| 0x0012 TC1 (High) | R | Bit 15 | Bit 14 | Bit 13 | Bit 12 | Bit 11 | Bit 10 | Bit 9 | Bit 8 |
| | W | | | | | | | | |
| 0x0013 TC1 (Low) | R | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
| | W | | | | | | | | |
| 0x0014 TC2 (High) | R | Bit 15 | Bit 14 | Bit 13 | Bit 12 | Bit 11 | Bit 10 | Bit 9 | Bit 8 |
| | W | | | | | | | | |
| 0x0015 TC2 (Low) | R | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
| | W | | | | | | | | |
| 0x0016 TC3 (High) | R | Bit 15 | Bit 14 | Bit 13 | Bit 12 | Bit 11 | Bit 10 | Bit 9 | Bit 8 |
| | W | | | | | | | | |
| 0x0017 TC3 (Low) | R | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
| | W | | | | | | | | |
| 0x0018 TC4 (High) | R | Bit 15 | Bit 14 | Bit 13 | Bit 12 | Bit 11 | Bit 10 | Bit 9 | Bit 8 |
| | W | | | | | | | | |
| 0x0019 TC4 (Low) | R | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
| | W | | | | | | | | |
| 0x001A TC5 (High) | R | Bit 15 | Bit 14 | Bit 13 | Bit 12 | Bit 11 | Bit 10 | Bit 9 | Bit 8 |
| | W | | | | | | | | |
| 0x001B TC5 (Low) | R | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
| | W | | | | | | | | |

= Unimplemented or Reserved

**Figure 14-2. ECT Register Summary (Sheet 2 of 5)**

| Register Name | | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|---|
| 0x001C TC6 (High) | R | Bit 15 | Bit 14 | Bit 13 | Bit 12 | Bit 11 | Bit 10 | Bit 9 | Bit 8 |
| | W | | | | | | | | |
| 0x001D TC6 (Low) | R | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
| | W | | | | | | | | |
| 0x001E TC7 (High) | R | Bit 15 | Bit 14 | Bit 13 | Bit 12 | Bit 11 | Bit 10 | Bit 9 | Bit 8 |
| | W | | | | | | | | |
| 0x001F TC7 (Low) | R | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
| | W | | | | | | | | |
| 0x0020 PACTL | R | 0 | PAEN | PAMOD | PEDGE | CLK1 | CLK0 | PA0VI | PAI |
| | W | | | | | | | | |
| 0x0021 PAFLG | R | 0 | 0 | 0 | 0 | 0 | 0 | PA0VF | PAIF |
| | W | | | | | | | | |
| 0x0022 PACN3 | R | PACNT7(15) | PACNT6(14) | PACNT5(13) | PACNT4(12) | PACNT3(11) | PACNT2(10) | PACNT1(9) | PACNT0(8) |
| | W | | | | | | | | |
| 0x0023 PACN2 | R | PACNT7 | PACNT6 | PACNT5 | PACNT4 | PACNT3 | PACNT2 | PACNT1 | PACNT0 |
| | W | | | | | | | | |
| 0x0024 PACN1 | R | PACNT7(15) | PACNT6(14) | PACNT5(13) | PACNT4(12) | PACNT3(11) | PACNT2(10) | PACNT1(9) | PACNT0(8) |
| | W | | | | | | | | |
| 0x0025 PACN0 | R | PACNT7 | PACNT6 | PACNT5 | PACNT4 | PACNT3 | PACNT2 | PACNT1 | PACNT0 |
| | W | | | | | | | | |
| 0x0026 MCCTL | R | MCZI | MODMC | RDMCL | 0 | 0 | MCEN | MCPR1 | MCPR0 |
| | W | | | | ICLAT | FLMC | | | |
| 0x0027 MCFLG | R | MCZF | 0 | 0 | 0 | POLF3 | POLF2 | POLF1 | POLF0 |
| | W | | | | | | | | |
| 0x0028 ICPAR | R | 0 | 0 | 0 | 0 | PA3EN | PA2EN | PA1EN | PA0EN |
| | W | | | | | | | | |
| 0x0029 DLYCT | R | DLY7 | DLY6 | DLY5 | DLY4 | DLY3 | DLY2 | DLY1 | DLY0 |
| | W | | | | | | | | |
| 0x002A ICOVW | R | NOVW7 | NOVW6 | NOVW5 | NOVW4 | NOVW3 | NOVW2 | NOVW1 | NOVW0 |
| | W | | | | | | | | |

= Unimplemented or Reserved

**Figure 14-2. ECT Register Summary (Sheet 3 of 5)**

| Register Name | | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|---|
| 0x002B ICSYS | R | SH37 | SH26 | SH15 | SH04 | TFMOD | PACMX | BUFEN | LATQ |
| | W | | | | | | | | |
| 0x002C OCPD | R | OCPD7 | OCPD6 | OCPD5 | OCPD4 | OCPD3 | OCPD2 | OCPD1 | OCPD0 |
| | W | | | | | | | | |
| 0x002D TIMTST | R | Timer Test Register | | | | | | | |
| | W | | | | | | | | |
| 0x002E PTPSR | R | PTPS7 | PTPS6 | PTPS5 | PTPS4 | PTPS3 | PTPS2 | PTPS1 | PTPS0 |
| | W | | | | | | | | |
| 0x002F PTMCPSR | R | PTMPS7 | PTMPS6 | PTMPS5 | PTMPS4 | PTMPS3 | PTMPS2 | PTMPS1 | PTMPS0 |
| | W | | | | | | | | |
| 0x0030 PBCTL | R | 0 | PBEN | 0 | 0 | 0 | 0 | PBOVI | 0 |
| | W | | | | | | | | |
| 0x0031 PBFLG | R | 0 | 0 | 0 | 0 | 0 | 0 | PBOVF | 0 |
| | W | | | | | | | | |
| 0x0032 PA3H | R | PA3H7 | PA3H6 | PA3H5 | PA3H4 | PA3H3 | PA3H2 | PA3H1 | PA3H0 |
| | W | | | | | | | | |
| 0x0033 PA2H | R | PA2H7 | PA2H6 | PA2H5 | PA2H4 | PA2H3 | PA2H2 | PA2H1 | PA2H0 |
| | W | | | | | | | | |
| 0x0034 PA1H | R | PA1H7 | PA1H6 | PA1H5 | PA1H4 | PA1H3 | PA1H2 | PA1H1 | PA1H0 |
| | W | | | | | | | | |
| 0x0035 PA0H | R | PA0H7 | PA0H6 | PA0H5 | PA0H4 | PA0H3 | PA0H2 | PA0H1 | PA0H0 |
| | W | | | | | | | | |
| 0x0036 MCCNT (High) | R | MCCNT15 | MCCNT14 | MCCNT13 | MCCNT12 | MCCNT11 | MCCNT10 | MCCNT9 | MCCNT8 |
| | W | | | | | | | | |
| 0x0037 MCCNT (Low) | R | MCCNT7 | MCCNT6 | MCCNT5 | MCCNT4 | MCCNT3 | MCCNT2 | MCCNT1 | MCCNT0 |
| | W | | | | | | | | |
| 0x0038 TC0H (High) | R | TC15 | TC14 | TC13 | TC12 | TC11 | TC10 | TC9 | TC8 |
| | W | | | | | | | | |
| 0x0039 TC0H (Low) | R | TC7 | TC6 | TC5 | TC4 | TC3 | TC2 | TC1 | TC0 |
| | | | | | | | | | |
| | | = Unimplemented or Reserved | | | | | | | |

**Figure 14-2. ECT Register Summary (Sheet 4 of 5)**

| Register Name | | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|---|
| 0x003A TC1H (High) | R | TC15 | TC14 | TC13 | TC12 | TC11 | TC10 | TC9 | TC8 |
| | W | | | | | | | | |
| 0x003B TC1H (Low) | R | TC7 | TC6 | TC5 | TC4 | TC3 | TC2 | TC1 | TC0 |
| | W | | | | | | | | |
| 0x003C TC2H (High) | R | TC15 | TC14 | TC13 | TC12 | TC11 | TC10 | TC9 | TC8 |
| | W | | | | | | | | |
| 0x003D TC2H (Low) | R | TC7 | TC6 | TC5 | TC4 | TC3 | TC2 | TC1 | TC0 |
| | W | | | | | | | | |
| 0x003E TC3H (High) | R | TC15 | TC14 | TC13 | TC12 | TC11 | TC10 | TC9 | TC8 |
| | W | | | | | | | | |
| 0x003F TC3H (Low) | R | TC7 | TC6 | TC5 | TC4 | TC3 | TC2 | TC1 | TC0 |
| | W | | | | | | | | |

= Unimplemented or Reserved

**Figure 14-2. ECT Register Summary (Sheet 5 of 5)**

### 14.3.2.1 Timer Input Capture/Output Compare Select Register (TIOS)

Module Base + 0x0000

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | IOS7 | IOS6 | IOS5 | IOS4 | IOS3 | IOS2 | IOS1 | IOS0 |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 14-3. Timer Input Capture/Output Compare Register (TIOS)**

Read or write: Anytime

All bits reset to zero.

**Table 14-2. TIOS Field Descriptions**

| Field | Description |
|---|---|
| 7:0<br>IOS[7:0] | **Input Capture or Output Compare Channel Configuration**<br>0 The corresponding channel acts as an input capture.<br>1 The corresponding channel acts as an output compare. |

### 14.3.2.2 Timer Compare Force Register (CFORC)

Module Base + 0x0001

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| W | FOC7 | FOC6 | FOC5 | FOC4 | FOC3 | FOC2 | FOC1 | FOC0 |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 14-4. Timer Compare Force Register (CFORC)**

Read or write: Anytime but reads will always return 0x0000 (1 state is transient).

All bits reset to zero.

**Table 14-3. CFORC Field Descriptions**

| Field | Description |
|---|---|
| 7:0<br>FOC[7:0] | **Force Output Compare Action for Channel 7:0** — A write to this register with the corresponding data bit(s) set causes the action which is programmed for output compare “x” to occur immediately. The action taken is the same as if a successful comparison had just taken place with the TCx register except the interrupt flag does not get set.<br>**Note:** A channel 7 event, which can be a counter overflow when TTOV[7] is set or A successful channel 7 output compare overrides any channel 6:0 compares. If a forced output compare on any channel occurs at the same time as the successful output compare, then the forced output compare action will take precedence and the interrupt flag will not get set. |

### 14.3.2.3 Output Compare 7 Mask Register (OC7M)

Module Base + 0x0002

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | OC7M7 | OC7M6 | OC7M5 | OC7M4 | OC7M3 | OC7M2 | OC7M1 | OC7M0 |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 14-5. Output Compare 7 Mask Register (OC7M)**

Read or write: Anytime

All bits reset to zero.

**Table 14-4. OC7M Field Descriptions**

| Field | Description |
|---|---|
| 7:0<br>OC7M[7:0] | **Output Compare Mask Action for Channel 7:0**<br>**A channel 7 event, which can be a counter overflow when TTOV[7] is set or a successful output compare on channel 7, overrides any channel 6:0 compares. For each OC7M bit that is set,the output compare action reflects the corresponding OC7D bit.**<br>0 The corresponding OC7Dx bit in the output compare 7 data register will not be transferred to the timer port on a channel 7 event, even if the corresponding pin is setup for output compare.<br>1 The corresponding OC7Dx bit in the output compare 7 data register will be transferred to the timer port on a channel 7 event.<br>**Note:** The corresponding channel must also be setup for output compare (IOSx = 1 andOCPDx = 0) for data to be transferred from the output compare 7 data register to the timer port. |

### 14.3.2.4 Output Compare 7 Data Register (OC7D)

Module Base + 0x0003

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | OC7D7 | OC7D6 | OC7D5 | OC7D4 | OC7D3 | OC7D2 | OC7D1 | OC7D0 |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 14-6. Output Compare 7 Data Register (OC7D)**

Read or write: Anytime

All bits reset to zero.

**Table 14-5. OC7D Field Descriptions**

| Field | Description |
|---|---|
| 7:0<br>OC7D[7:0] | **Output Compare 7 Data Bits** — A channel 7 event, which can be a counter overflow when TTOV[7] is set or A channel 7 output compare can cause bits in the output compare 7 data register to transfer to the timer port data register depending on the output compare 7 mask register. |

### 14.3.2.5 Timer Count Register (TCNT)

Module Base + 0x0004

| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | TCNT15 | TCNT14 | TCNT13 | TCNT12 | TCNT11 | TCNT10 | TCNT9 | TCNT8 |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 14-7. Timer Count Register High (TCNT)**

Module Base + 0x0005

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | TCNT7 | TCNT6 | TCNT5 | TCNT4 | TCNT3 | TCNT2 | TCNT1 | TCNT0 |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 14-8. Timer Count Register Low (TCNT)**

Read: Anytime

Write: Writable in special modes.

All bits reset to zero.

**Table 14-6. TCNT Field Descriptions**

| Field | Description |
|---|---|
| 15:0<br>TCNT[15:0] | **Timer Counter Bits** — The 16-bit main timer is an up counter. A read to this register will return the current value of the counter. Access to the counter register will take place in one clock cycle.<br>**Note:** A separate read/write for high byte and low byte in test mode will give a different result than accessing them as a word. The period of the first count after a write to the TCNT registers may be a different size because the write is not synchronized with the prescaler clock. |

## 14.3.2.6 Timer System Control Register 1 (TSCR1)

Module Base + 0x0006

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | TEN | TSWAI | TSFRZ | TFFCA | PRNT | 0 | 0 | 0 |
| W | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented or Reserved

**Figure 14-9. Timer System Control Register 1 (TSCR1)**

Read or write: Anytime except PRNT bit is write once

All bits reset to zero.

**Table 14-7. TSCR1 Field Descriptions**

| Field | Description |
|---|---|
| 7<br>TEN | **Timer Enable**<br>0 Disables the main timer, including the counter. Can be used for reducing power consumption.<br>1 Allows the timer to function normally.<br>**Note:** If for any reason the timer is not active, there is no ÷64 clock for the pulse accumulator since the ÷64 is generated by the timer prescaler. |
| 6<br>TSWAI | **Timer Module Stops While in Wait**<br>0 Allows the timer module to continue running during wait.<br>1 Disables the timer counter, pulse accumulators and modulus down counter when the MCU is in wait mode. Timer interrupts cannot be used to get the MCU out of wait. |

**Table 14-7. TSCR1 Field Descriptions (continued)**

| Field | Description |
|---|---|
| 5<br>TSFRZ | **Timer and Modulus Counter Stop While in Freeze Mode**<br>0 Allows the timer and modulus counter to continue running while in freeze mode.<br>1 Disables the timer and modulus counter whenever the MCU is in freeze mode. This is useful for emulation. The pulse accumulators do not stop in freeze mode. |
| 4<br>TFFCA | **Timer Fast Flag Clear All**<br>0 Allows the timer flag clearing to function normally.<br>1 A read from an input capture or a write to the output compare channel registers causes the corresponding channel flag, CxF, to be cleared in the TFLG1 register. Any access to the TCNT register clears the TOF flag in the TFLG2 register. Any access to the PACN3 and PACN2 registers clears the PAOVF and PAIF flags in the PAFLG register. Any access to the PACN1 and PACN0 registers clears the PBOVF flag in the PBFLG register. Any access to the MCCNT register clears the MCZF flag in the MCFLG register. This has the advantage of eliminating software overhead in a separate clear sequence. Extra care is required to avoid accidental flag clearing due to unintended accesses.<br>**Note:** The flags cannot be cleared via the normal flag clearing mechanism (writing a one to the flag) when TFFCA = 1. |
| 3<br>PRNT | **Precision Timer**<br>0 Enables legacy timer. Only bits DLY0 and DLY1 of the DLYCT register are used for the delay selection of the delay counter. PR0, PR1, and PR2 bits of the TSCR2 register are used for timer counter prescaler selection. MCPR0 and MCPR1 bits of the MCCTL register are used for modulus down counter prescaler selection.<br>1 Enables precision timer. All bits in the DLYCT register are used for the delay selection, all bits of the PTPSR register are used for Precision Timer Prescaler Selection, and all bits of PTMCPSR register are used for the prescaler Precision Timer Modulus Counter Prescaler selection. |

### 14.3.2.7 Timer Toggle On Overflow Register 1 (TTOV)

Module Base + 0x0007

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | TOV7 | TOV6 | TOV5 | TOV4 | TOV3 | TOV2 | TOV1 | TOV0 |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 14-10. Timer Toggle On Overflow Register 1 (TTOV)**

Read or write: Anytime

All bits reset to zero.

**Table 14-8. TTOV Field Descriptions**

| Field | Description |
|---|---|
| 7:0<br>TOV[7:0] | **Toggle On Overflow Bits** — TOV97:0] toggles output compare pin on timer counter overflow. This feature only takes effect when in output compare mode. When set, it takes precedence over forced output compare but not channel 7 override events.<br>0 Toggle output compare pin on overflow feature disabled.<br>1 Toggle output compare pin on overflow feature enabled. |

### 14.3.2.8 Timer Control Register 1/Timer Control Register 2 (TCTL1/TCTL2)

Module Base + 0x0008

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | OM7 | OL7 | OM6 | OL6 | OM5 | OL5 | OM4 | OL4 |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 14-11. Timer Control Register 1 (TCTL1)**

Module Base + 0x0009

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | OM3 | OL3 | OM2 | OL2 | OM1 | OL1 | OM0 | OL0 |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 14-12. Timer Control Register 2 (TCTL2)**

Read or write: Anytime

All bits reset to zero.

**Table 14-9. TCTL1/TCTL2 Field Descriptions**

| Field | Description |
|---|---|
| OM[7:0]<br>7, 5, 3, 1 | OMx — Output Mode<br>OLx — Output Level<br>These eight pairs of control bits are encoded to specify the output action to be taken as a result of a successful OCx compare. When either OMx or OLx is one, the pin associated with OCx becomes an output tied to OCx. See Table 14-10. |
| OL[7:0]<br>6, 4, 2, 0 | |

**Table 14-10. Compare Result Output Action**

| OMx | OLx | Action |
|---|---|---|
| 0 | 0 | No output compare<br>action on the timer output signal |
| 0 | 1 | Toggle OCx output line |
| 1 | 0 | Clear OCx output line to zero |
| 1 | 1 | Set OCx output line to one |

**NOTE**

To enable output action by OMx and OLx bits on timer port, the corresponding bit in OC7M should be cleared. The settings for these bits can be seen in Table 14-11

**Table 14-11. The OC7 and OCx event priority**

| OC7M7=0 | | | | OC7M7=1 | | | |
|---|---|---|---|---|---|---|---|
| OC7Mx=1 | | OC7Mx=0 | | OC7Mx=1 | | OC7Mx=0 | |
| TC7=TCx | TC7>TCx | TC7=TCx | TC7>TCx | TC7=TCx | TC7>TCx | TC7=TCx | TC7>TCx |
| IOCx=OC7Dx IOC7=OM7/OL7 | IOCx=OC7Dx +OMx/OLx IOC7=OM7/OL7 | IOCx=OMx/OLx IOC7=OM7/OL7 | | IOCx=OC7Dx IOC7=OC7D7 | IOCx=OC7Dx +OMx/OLx IOC7=OC7D7 | IOCx=OMx/OLx IOC7=OC7D7 | |

Note: in Table 14-11,the IOS7 and IOSx should be set to 1

IOSx is the register TIOS bit x,

OC7Mx is the register OC7M bit x,

TCx is timer Input Capture/Output Compare register,

IOCx is channel x,

OMx/OLx is the register TCTL1/TCTL2,

OC7Dx is the register OC7D bit x.

IOCx = OC7Dx+ OMx/OLx, means that both OC7 event and OCx event will change channel x value.

### 14.3.2.9 Timer Control Register 3/Timer Control Register 4 (TCTL3/TCTL4)

Module Base + 0x000A

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | EDG7B | EDG7A | EDG6B | EDG6A | EDG5B | EDG5A | EDG4B | EDG4A |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 14-13. Timer Control Register 3 (TCTL3)**

Module Base + 0x000B

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | EDG3B | EDG3A | EDG2B | EDG2A | EDG1B | EDG1A | EDG0B | EDG0A |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 14-14. Timer Control Register 4 (TCTL4)**

Read or write: Anytime

All bits reset to zero.

**Table 14-12. TCTL3/TCTL4 Field Descriptions**

| Field | Description |
|---|---|
| EDG[7:0]B<br>7, 5, 3, 1<br>EDG[7:0]A<br>6, 4, 2, 0 | **Input Capture Edge Control** — These eight pairs of control bits configure the input capture edge detector circuits for each input capture channel. The four pairs of control bits in TCTL4 also configure the input capture edge control for the four 8-bit pulse accumulators PAC0–PAC3.EDG0B and EDG0A in TCTL4 also determine the active edge for the 16-bit pulse accumulator PACB. See Table 14-13. |

**Table 14-13. Edge Detector Circuit Configuration**

| EDGxB | EDGxA | Configuration |
|---|---|---|
| 0 | 0 | Capture disabled |
| 0 | 1 | Capture on rising edges only |
| 1 | 0 | Capture on falling edges only |
| 1 | 1 | Capture on any edge (rising or falling) |

### 14.3.2.10 Timer Interrupt Enable Register (TIE)

Module Base + 0x000C

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | C7I | C6I | C5I | C4I | C3I | C2I | C1I | C0I |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 14-15. Timer Interrupt Enable Register (TIE)**

Read or write: Anytime

All bits reset to zero.

The bits C7I–C0I correspond bit-for-bit with the flags in the TFLG1 status register.

**Table 14-14. TIE Field Descriptions**

| Field | Description |
|---|---|
| 7:0<br>C[7:0]I | **Input Capture/Output Compare “x” Interrupt Enable**<br>0 The corresponding flag is disabled from causing a hardware interrupt.<br>1 The corresponding flag is enabled to cause an interrupt. |

### 14.3.2.11 Timer System Control Register 2 (TSCR2)

Module Base + 0x000D

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | TOI | 0 | 0 | 0 | TCRE | PR2 | PR1 | PR0 |
| W | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented or Reserved

**Figure 14-16. Timer System Control Register 2 (TSCR2)**

Read or write: Anytime

All bits reset to zero.

**Table 14-15. TSCR2 Field Descriptions**

| Field | Description |
|---|---|
| 7<br>TOI | **Timer Overflow Interrupt Enable**<br>0 Timer overflow interrupt disabled.<br>1 Hardware interrupt requested when TOF flag set. |
| 3<br>TCRE | **Timer Counter Reset Enable** — This bit allows the timer counter to be reset by a successful channel 7 output compare. This mode of operation is similar to an up-counting modulus counter.<br>0 Counter reset disabled and counter free runs.<br>1 Counter reset by a successful output compare on channel 7.<br><br>**Note:** If register TC7 = 0x0000 and TCRE = 1, then the TCNT register will stay at 0x0000 continuously. If register TC7 = 0xFFFF and TCRE = 1, the TOF flag will never be set when TCNT is reset from 0xFFFF to 0x0000.<br>**Note:** TCRE=1 and TC7!=0, the TCNT cycle period will be TC7 x "prescaler counter width" + "1 Bus Clock". When TCRE is set and TC7 is not equal to 0, TCNT will cycle from 0 to TC7. When TCNT reaches TC7 value, it will last only one bus cycle then reset to 0. for a more detail explanation please refer to Figure 14-17.<br>**Note:** in Figure 14-17,if PR[2:0] is equal to 0, one prescaler counter equal to one bus clock |
| 2:0<br>PR[2:0] | **Timer Prescaler Select** — These three bits specify the division rate of the main Timer prescaler when the PRNT bit of register TSCR1 is set to 0. The newly selected prescale factor will not take effect until the next synchronized edge where all prescale counter stages equal zero. See Table 14-16. |

**Figure 14-17. The TCNT cycle diagram under TCRE=1 condition**

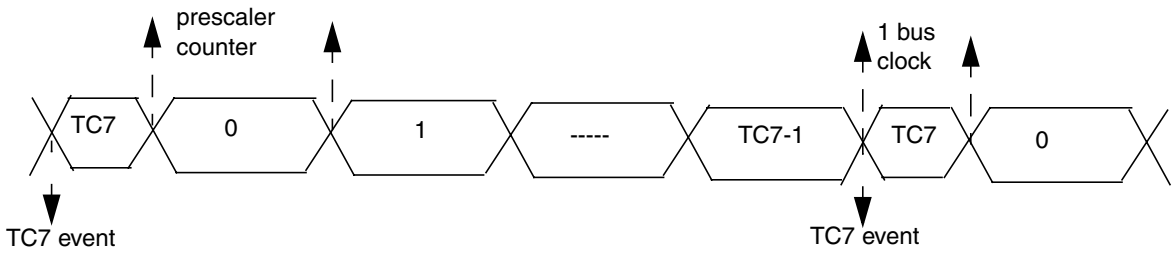



**Table 14-16. Prescaler Selection**

| PR2 | PR1 | PR0 | Prescale Factor |
|---|---|---|---|
| 0 | 0 | 0 | 1 |
| 0 | 0 | 1 | 2 |
| 0 | 1 | 0 | 4 |
| 0 | 1 | 1 | 8 |
| 1 | 0 | 0 | 16 |
| 1 | 0 | 1 | 32 |
| 1 | 1 | 0 | 64 |
| 1 | 1 | 1 | 128 |

### 14.3.2.12 Main Timer Interrupt Flag 1 (TFLG1)

Module Base + 0x000E

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | C7F | C6F | C5F | C4F | C3F | C2F | C1F | C0F |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 14-18. Main Timer Interrupt Flag 1 (TFLG1)**

Read: Anytime

Write used in the flag clearing mechanism. Writing a one to the flag clears the flag. Writing a zero will not affect the current status of the bit.

**NOTE**

When TFFCA = 1, the flags cannot be cleared via the normal flag clearing mechanism (writing a one to the flag). Reference Section 14.3.2.6, “Timer System Control Register 1 (TSCR1)”.

All bits reset to zero.

TFLG1 indicates when interrupt conditions have occurred. The flags can be cleared via the normal flag clearing mechanism (writing a one to the flag) or via the fast flag clearing mechanism (reference TFFCA bit in Section 14.3.2.6, “Timer System Control Register 1 (TSCR1)”).

Use of the TFMOD bit in the ICSYS register in conjunction with the use of the ICOVW register allows a timer interrupt to be generated after capturing two values in the capture and holding registers, instead of generating an interrupt for every capture.

**Table 14-17. TFLG1 Field Descriptions**

| Field | Description |
|---|---|
| 7:0<br>C[7:0]F | **Input Capture/Output Compare Channel “x” Flag** — A CxF flag is set when a corresponding input capture or output compare is detected. C0F can also be set by 16-bit Pulse Accumulator B (PACB). C3F–C0F can also be set by 8-bit pulse accumulators PAC3–PAC0.<br>If the delay counter is enabled, the CxF flag will not be set until after the delay. |

### 14.3.2.13 Main Timer Interrupt Flag 2 (TFLG2)

Module Base + 0x000F

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | TOF | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| W | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented or Reserved

**Figure 14-19. Main Timer Interrupt Flag 2 (TFLG2)**

Read: Anytime

Write used in the flag clearing mechanism. Writing a one to the flag clears the flag. Writing a zero will not affect the current status of the bit.

**NOTE**

When TFFCA = 1, the flag cannot be cleared via the normal flag clearing mechanism (writing a one to the flag). Reference Section 14.3.2.6, “Timer System Control Register 1 (TSCR1)”.

All bits reset to zero.

TFLG2 indicates when interrupt conditions have occurred. The flag can be cleared via the normal flag clearing mechanism (writing a one to the flag) or via the fast flag clearing mechanism (Reference TFFCA bit in Section 14.3.2.6, “Timer System Control Register 1 (TSCR1)”).

**Table 14-18. TFLG2 Field Descriptions**

| Field | Description |
|---|---|
| 7<br>TOF | **Timer Overflow Flag** — Set when 16-bit free-running timer overflows from 0xFFFF to 0x0000. |

### 14.3.2.14 Timer Input Capture/Output Compare Registers 0–7

Module Base + 0x0010

| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | Bit 15 | Bit 14 | Bit 13 | Bit 12 | Bit 11 | Bit 10 | Bit 9 | Bit 8 |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 14-20. Timer Input Capture/Output Compare Register 0 High (TC0)**

Module Base + 0x0011

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 14-21. Timer Input Capture/Output Compare Register 0 Low (TC0)**

Module Base + 0x0012

| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | Bit 15 | Bit 14 | Bit 13 | Bit 12 | Bit 11 | Bit 10 | Bit 9 | Bit 8 |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 14-22. Timer Input Capture/Output Compare Register 1 High (TC1)**

Module Base + 0x0013

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 14-23. Timer Input Capture/Output Compare Register 1 Low (TC1)**

Module Base + 0x0014

| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | Bit 15 | Bit 14 | Bit 13 | Bit 12 | Bit 11 | Bit 10 | Bit 9 | Bit 8 |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 14-24. Timer Input Capture/Output Compare Register 2 High (TC2)**

Module Base + 0x0015

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 14-25. Timer Input Capture/Output Compare Register 2 Low (TC2)**

Module Base + 0x0016

| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | Bit 15 | Bit 14 | Bit 13 | Bit 12 | Bit 11 | Bit 10 | Bit 9 | Bit 8 |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 14-26. Timer Input Capture/Output Compare Register 3 High (TC3)**

Module Base + 0x0017

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 14-27. Timer Input Capture/Output Compare Register 3 Low (TC3)**

Module Base + 0x0018

| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | Bit 15 | Bit 14 | Bit 13 | Bit 12 | Bit 11 | Bit 10 | Bit 9 | Bit 8 |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 14-28. Timer Input Capture/Output Compare Register 4 High (TC4)**

Module Base + 0x0019

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 14-29. Timer Input Capture/Output Compare Register 4 Low (TC4)**

Module Base + 0x001A

| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | Bit 15 | Bit 14 | Bit 13 | Bit 12 | Bit 11 | Bit 10 | Bit 9 | Bit 8 |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 14-30. Timer Input Capture/Output Compare Register 5 High (TC5)**

Module Base + 0x001B

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 14-31. Timer Input Capture/Output Compare Register 5 Low (TC5)**

Module Base + 0x001C

| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | Bit 15 | Bit 14 | Bit 13 | Bit 12 | Bit 11 | Bit 10 | Bit 9 | Bit 8 |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 14-32. Timer Input Capture/Output Compare Register 6 High (TC6)**

Module Base + 0x001D

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 14-33. Timer Input Capture/Output Compare Register 6 Low (TC6)**

Module Base + 0x001E

| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | Bit 15 | Bit 14 | Bit 13 | Bit 12 | Bit 11 | Bit 10 | Bit 9 | Bit 8 |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 14-34. Timer Input Capture/Output Compare Register 7 High (TC7)**

Module Base + 0x001F

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 14-35. Timer Input Capture/Output Compare Register 7 Low (TC7)**

Read: Anytime

Write anytime for output compare function. Writes to these registers have no meaning or effect during input capture.

All bits reset to zero.

Depending on the TIOS bit for the corresponding channel, these registers are used to latch the value of the free-running counter when a defined transition is sensed by the corresponding input capture edge detector or to trigger an output action for output compare.

### 14.3.2.15 16-Bit Pulse Accumulator A Control Register (PACTL)

Module Base + 0x0020

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | 0 | PAEN | PAMOD | PEDGE | CLK1 | CLK0 | PAOVI | PAI |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented or Reserved

**Figure 14-36. 16-Bit Pulse Accumulator Control Register (PACTL)**

Read: Anytime

Write: Anytime

All bits reset to zero.

**Table 14-19. PACTL Field Descriptions**

| Field | Description |
|---|---|
| 6<br>PAEN | **Pulse Accumulator A System Enable** — PAEN is independent from TEN. With timer disabled, the pulse accumulator can still function unless pulse accumulator is disabled.<br>0 16-Bit Pulse Accumulator A system disabled. 8-bit PAC3 and PAC2 can be enabled when their related enable bits in ICPAR are set. Pulse Accumulator Input Edge Flag (PAIF) function is disabled.<br>1 16-Bit Pulse Accumulator A system enabled. The two 8-bit pulse accumulators PAC3 and PAC2 are cascaded to form the PACA 16-bit pulse accumulator. When PACA in enabled, the PACN3 and PACN2 registers contents are respectively the high and low byte of the PACA. PA3EN and PA2EN control bits in ICPAR have no effect. Pulse Accumulator Input Edge Flag (PAIF) function is enabled. The PACA shares the input pin with IC7. |
| 5<br>PAMOD | **Pulse Accumulator Mode** — This bit is active only when the Pulse Accumulator A is enabled (PAEN = 1).<br>0 Event counter mode<br>1 Gated time accumulation mode |

**Table 14-19. PACTL Field Descriptions (continued)**

| Field | Description |
|---|---|
| 4<br>PEDGE | **Pulse Accumulator Edge Control** — This bit is active only when the Pulse Accumulator A is enabled (PAEN = 1). Refer to Table 14-20.<br>For PAMOD bit = 0 (event counter mode).<br>0 Falling edges on IC7 pin cause the count to be incremented<br>1 Rising edges on IC7 pin cause the count to be incremented<br>For PAMOD bit = 1 (gated time accumulation mode).<br>0 IC7 input pin high enables bus clock divided by 64 to Pulse Accumulator and the trailing falling edge on IC7 sets the PAIF flag.<br>1 IC7 input pin low enables bus clock divided by 64 to Pulse Accumulator and the trailing rising edge on IC7 sets the PAIF flag.<br>If the timer is not active (TEN = 0 in TSCR1), there is no divide-by-64 since the ÷64 clock is generated by the timer prescaler. |
| 3:2<br>CLK[1:0] | **Clock Select Bits** — For the description of PACLK please refer to Figure 14-72.<br>If the pulse accumulator is disabled (PAEN = 0), the prescaler clock from the timer is always used as an input clock to the timer counter. The change from one selected clock to the other happens immediately after these bits are written. Refer to Table 14-21. |
| 2<br>PAOVI | **Pulse Accumulator A Overflow Interrupt Enable**<br>0 Interrupt inhibited<br>1 Interrupt requested if PAOVF is set |
| 0<br>PAI | **Pulse Accumulator Input Interrupt Enable**<br>0 Interrupt inhibited<br>1 Interrupt requested if PAIF is set |

**Table 14-20. Pin Action**

| PAMOD | PEDGE | Pin Action |
|---|---|---|
| 0 | 0 | Falling edge |
| 0 | 1 | Rising edge |
| 1 | 0 | Divide by 64 clock enabled with pin high level |
| 1 | 1 | Divide by 64 clock enabled with pin low level |

**Table 14-21. Clock Selection**

| CLK1 | CLK0 | Clock Source |
|---|---|---|
| 0 | 0 | Use timer prescaler clock as timer counter clock |
| 0 | 1 | Use PACLK as input to timer counter clock |
| 1 | 0 | Use PACLK/256 as timer counter clock frequency |
| 1 | 1 | Use PACLK/65536 as timer counter clock frequency |

### 14.3.2.16 Pulse Accumulator A Flag Register (PAFLG)

Module Base + 0x0021

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | 0 | 0 | 0 | 0 | PAOVF | PAIF |
| W | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented or Reserved

**Figure 14-37. Pulse Accumulator A Flag Register (PAFLG)**

Read: Anytime

Write used in the flag clearing mechanism. Writing a one to the flag clears the flag. Writing a zero will not affect the current status of the bit.

**NOTE**

When TFFCA = 1, the flags cannot be cleared via the normal flag clearing mechanism (writing a one to the flag). Reference Section 14.3.2.6, “Timer System Control Register 1 (TSCR1)”.

All bits reset to zero.

PAFLG indicates when interrupt conditions have occurred. The flags can be cleared via the normal flag clearing mechanism (writing a one to the flag) or via the fast flag clearing mechanism (Reference TFFCA bit in Section 14.3.2.6, “Timer System Control Register 1 (TSCR1)”).

**Table 14-22. PAFLG Field Descriptions**

| Field | Description |
|---|---|
| 1<br>PAOVF | **Pulse Accumulator A Overflow Flag** — Set when the 16-bit pulse accumulator A overflows from 0xFFFF to 0x0000, or when 8-bit pulse accumulator 3 (PAC3) overflows from 0x00FF to 0x0000.<br>When PACMX = 1, PAOVF bit can also be set if 8-bit pulse accumulator 3 (PAC3) reaches 0x00FF followed by an active edge on IC3. |
| 0<br>PAIF | **Pulse Accumulator Input edge Flag** — Set when the selected edge is detected at the IC7 input pin. In event mode the event edge triggers PAIF and in gated time accumulation mode the trailing edge of the gate signal at the IC7 input pin triggers PAIF. |

### 14.3.2.17 Pulse Accumulators Count Registers (PACN3 and PACN2)

Module Base + 0x0022

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | PACNT7(15) | PACNT6(14) | PACNT5(13) | PACNT4(12) | PACNT3(11) | PACNT2(10) | PACNT1(9) | PACNT0(8) |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 14-38. Pulse Accumulators Count Register 3 (PACN3)**

Module Base + 0x0023

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | PACNT7 | PACNT6 | PACNT5 | PACNT4 | PACNT3 | PACNT2 | PACNT1 | PACNT0 |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 14-39. Pulse Accumulators Count Register 2 (PACN2)**

Read: Anytime

Write: Anytime

All bits reset to zero.

The two 8-bit pulse accumulators PAC3 and PAC2 are cascaded to form the PACA 16-bit pulse accumulator. When PACA in enabled (PAEN = 1 in PACTL), the PACN3 and PACN2 registers contents are respectively the high and low byte of the PACA.

When PACN3 overflows from 0x00FF to 0x0000, the interrupt flag PAOVF in PAFLG is set.

Full count register access will take place in one clock cycle.

**NOTE**

A separate read/write for high byte and low byte will give a different result than accessing them as a word.

When clocking pulse and write to the registers occurs simultaneously, write takes priority and the register is not incremented.

### 14.3.2.18 Pulse Accumulators Count Registers (PACN1 and PACN0)

Module Base + 0x0024

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | PACNT7(15) | PACNT6(14) | PACNT5(13) | PACNT4(12) | PACNT3(11) | PACNT2(10) | PACNT1(9) | PACNT0(8) |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 14-40. Pulse Accumulators Count Register 1 (PACN1)**

Module Base + 0x0025

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | PACNT7 | PACNT6 | PACNT5 | PACNT4 | PACNT3 | PACNT2 | PACNT1 | PACNT0 |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 14-41. Pulse Accumulators Count Register 0 (PACN0)**

Read: Anytime

Write: Anytime

All bits reset to zero.

The two 8-bit pulse accumulators PAC1 and PAC0 are cascaded to form the PACB 16-bit pulse accumulator. When PACB in enabled, (PBEN = 1 in PBCTL) the PACN1 and PACN0 registers contents are respectively the high and low byte of the PACB.

When PACN1 overflows from 0x00FF to 0x0000, the interrupt flag PBOVF in PBFLG is set.

Full count register access will take place in one clock cycle.

**NOTE**

A separate read/write for high byte and low byte will give a different result than accessing them as a word.

When clocking pulse and write to the registers occurs simultaneously, write takes priority and the register is not incremented.

### 14.3.2.19 16-Bit Modulus Down-Counter Control Register (MCCTL)

Module Base + 0x0026

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | MCZI | MODMC | RDMCL | 0 | 0 | MCEN | MCPR1 | MCPR0 |
| W | | | | ICLAT | FLMC | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 14-42. 16-Bit Modulus Down-Counter Control Register (MCCTL)**

Read: Anytime

Write: Anytime

All bits reset to zero.

**Table 14-23. MCCTL Field Descriptions**

| Field | Description |
|---|---|
| 7<br>MCZI | **Modulus Counter Underflow Interrupt Enable**<br>0 Modulus counter interrupt is disabled.<br>1 Modulus counter interrupt is enabled. |
| 6<br>MODMC | **Modulus Mode Enable**<br>0 The modulus counter counts down from the value written to it and will stop at 0x0000.<br>1 Modulus mode is enabled. When the modulus counter reaches 0x0000, the counter is loaded with the latest value written to the modulus count register.<br>**Note:** For proper operation, the MCEN bit should be cleared before modifying the MODMC bit in order to reset the modulus counter to 0xFFFF. |
| 5<br>RDMCL | **Read Modulus Down-Counter Load**<br>0 Reads of the modulus count register (MCCNT) will return the present value of the count register.<br>1 Reads of the modulus count register (MCCNT) will return the contents of the load register. |

**Table 14-23. MCCTL Field Descriptions (continued)**

| Field | Description |
|---|---|
| 4<br>ICLAT | **Input Capture Force Latch Action** — When input capture latch mode is enabled (LATQ and BUFEN bit in ICSYS are set), a write one to this bit immediately forces the contents of the input capture registers TC0 to TC3 and their corresponding 8-bit pulse accumulators to be latched into the associated holding registers. The pulse accumulators will be automatically cleared when the latch action occurs.<br>Writing zero to this bit has no effect. Read of this bit will always return zero. |
| 3<br>FLMC | **Force Load Register into the Modulus Counter Count Register** — This bit is active only when the modulus down-counter is enabled (MCEN = 1).<br>A write one into this bit loads the load register into the modulus counter count register (MCCNT). This also resets the modulus counter prescaler.<br>Write zero to this bit has no effect. Read of this bit will return always zero. |
| 2<br>MCEN | **Modulus Down-Counter Enable**<br>0 Modulus counter disabled. The modulus counter (MCCNT) is preset to 0xFFFF. This will prevent an early interrupt flag when the modulus down-counter is enabled.<br>1 Modulus counter is enabled. |
| 1:0<br>MCPR[1:0] | **Modulus Counter Prescaler Select** — These two bits specify the division rate of the modulus counter prescaler when PRNT of TSCR1 is set to 0. The newly selected prescaler division rate will not be effective until a load of the load register into the modulus counter count register occurs. |

**Table 14-24. Modulus Counter Prescaler Select**

| MCPR1 | MCPR0 | Prescaler Division |
|---|---|---|
| 0 | 0 | 1 |
| 0 | 1 | 4 |
| 1 | 0 | 8 |
| 1 | 1 | 16 |

### 14.3.2.20 16-Bit Modulus Down-Counter FLAG Register (MCFLG)

Module Base + 0x0027

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | MCZF | 0 | 0 | 0 | POLF3 | POLF2 | POLF1 | POLF0 |
| W | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented or Reserved

**Figure 14-43. 16-Bit Modulus Down-Counter FLAG Register (MCFLG)**

Read: Anytime

Write only used in the flag clearing mechanism for bit 7. Writing a one to bit 7 clears the flag. Writing a zero will not affect the current status of the bit.

**NOTE**

When TFFCA = 1, the flag cannot be cleared via the normal flag clearing mechanism (writing a one to the flag). Reference Section 14.3.2.6, “Timer System Control Register 1 (TSCR1)”.

All bits reset to zero.

**Table 14-25. MCFLG Field Descriptions**

| Field | Description |
|---|---|
| 7<br>MCZF | **Modulus Counter Underflow Flag** — The flag is set when the modulus down-counter reaches 0x0000.<br>The flag indicates when interrupt conditions have occurred. The flag can be cleared via the normal flag clearing mechanism (writing a one to the flag) or via the fast flag clearing mechanism (Reference TFFCA bit in Section 14.3.2.6, “Timer System Control Register 1 (TSCR1)”). |
| 3:0<br>POLF[3:0] | **First Input Capture Polarity Status** — These are read only bits. Writes to these bits have no effect.<br>Each status bit gives the polarity of the first edge which has caused an input capture to occur after capture latch has been read.<br>Each POLFx corresponds to a timer PORTx input.<br>0 The first input capture has been caused by a falling edge.<br>1 The first input capture has been caused by a rising edge. |

## 14.3.2.21 ICPAR — Input Control Pulse Accumulators Register (ICPAR)

Module Base + 0x0028

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | 0 | 0 | PA3EN | PA2EN | PA1EN | PA0EN |
| W | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented or Reserved

**Figure 14-44. Input Control Pulse Accumulators Register (ICPAR)**

Read: Anytime

Write: Anytime.

All bits reset to zero.

The 8-bit pulse accumulators PAC3 and PAC2 can be enabled only if PAEN in PACTL is cleared. If PAEN is set, PA3EN and PA2EN have no effect.

The 8-bit pulse accumulators PAC1 and PAC0 can be enabled only if PBEN in PBCTL is cleared. If PBEN is set, PA1EN and PA0EN have no effect.

**Table 14-26. ICPAR Field Descriptions**

| Field | Description |
|---|---|
| 3:0<br>PA[3:0]EN | **8-Bit Pulse Accumulator ‘x’ Enable**<br>0 8-Bit Pulse Accumulator is disabled.<br>1 8-Bit Pulse Accumulator is enabled. |

### 14.3.2.22 Delay Counter Control Register (DLYCT)

Module Base + 0x0029

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | DLY7 | DLY6 | DLY5 | DLY4 | DLY3 | DLY2 | DLY1 | DLY0 |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 14-45. Delay Counter Control Register (DLYCT)**

Read: Anytime

Write: Anytime

All bits reset to zero.

**Table 14-27. DLYCT Field Descriptions**

| Field | Description |
|---|---|
| 7:0<br>DLY[7:0] | **Delay Counter Select** — When the PRNT bit of TSCR1 register is set to 0, only bits DLY0, DLY1 are used to calculate the delay.Table 14-28 shows the delay settings in this case.<br>When the PRNT bit of TSCR1 register is set to 1, all bits are used to set a more precise delay. Table 14-29 shows the delay settings in this case. After detection of a valid edge on an input capture pin, the delay counter counts the pre-selected number of [(dly_cnt + 1)*4]bus clock cycles, then it will generate a pulse on its output if the level of input signal, after the preset delay, is the opposite of the level before the transition.This will avoid reaction to narrow input pulses.<br>Delay between two active edges of the input signal period should be longer than the selected counter delay.<br>**Note:** It is recommended to not write to this register while the timer is enabled, that is when TEN is set in register TSCR1. |

**Table 14-28. Delay Counter Select when PRNT = 0**

| DLY1 | DLY0 | Delay |
|---|---|---|
| 0 | 0 | Disabled |
| 0 | 1 | 256 bus clock cycles |
| 1 | 0 | 512 bus clock cycles |
| 1 | 1 | 1024 bus clock cycles |

**Table 14-29. Delay Counter Select Examples when PRNT = 1**

| DLY7 | DLY6 | DLY5 | DLY4 | DLY3 | DLY2 | DLY1 | DLY0 | Delay |
|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | Disabled (bypassed) |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 8 bus clock cycles |
| 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 12 bus clock cycles |
| 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 16 bus clock cycles |
| 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 20 bus clock cycles |
| 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 24 bus clock cycles |
| 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 28 bus clock cycles |

**Table 14-29. Delay Counter Select Examples when PRNT = 1**

| DLY7 | DLY6 | DLY5 | DLY4 | DLY3 | DLY2 | DLY1 | DLY0 | Delay |
|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 32 bus clock cycles |
| 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 64 bus clock cycles |
| 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 128 bus clock cycles |
| 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 256 bus clock cycles |
| 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 512 bus clock cycles |
| 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1024 bus clock cycles |

### 14.3.2.23 Input Control Overwrite Register (ICOVW)

Module Base + 0x002A

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | NOVW7 | NOVW6 | NOVW5 | NOVW4 | NOVW3 | NOVW2 | NOVW1 | NOVW0 |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 14-46. Input Control Overwrite Register (ICOVW)**

Read: Anytime

Write: Anytime

All bits reset to zero.

**Table 14-30. ICOVW Field Descriptions**

| Field | Description |
|---|---|
| 7:0<br>NOVW[7:0] | **No Input Capture Overwrite**<br>0 The contents of the related capture register or holding register can be overwritten when a new input capture or latch occurs.<br>1 The related capture register or holding register cannot be written by an event unless they are empty (see Section 14.4.1.1, “IC Channels”). This will prevent the captured value being overwritten until it is read or latched in the holding register. |

### 14.3.2.24 Input Control System Control Register (ICSYS)

Module Base + 0x002B

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | SH37 | SH26 | SH15 | SH04 | TFMOD | PACMX | BUFEN | LATQ |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 14-47. Input Control System Register (ICSYS)**

Read: Anytime

Write: Once in normal modes

All bits reset to zero.

**Table 14-31. ICSYS Field Descriptions**

| Field | Description |
|---|---|
| 7:4<br>SHxy | **Share Input action of Input Capture Channels x and y**<br>0 Normal operation<br>1 The channel input ‘x’ causes the same action on the channel ‘y’. The port pin ‘x’ and the corresponding edge detector is used to be active on the channel ‘y’. |
| 3<br>TFMOD | **Timer Flag Setting Mode** — Use of the TFMOD bit in conjunction with the use of the ICOVW register allows a timer interrupt to be generated after capturing two values in the capture and holding registers instead of generating an interrupt for every capture.<br>By setting TFMOD in queue mode, when NOVWx bit is set and the corresponding capture and holding registers are emptied, an input capture event will first update the related input capture register with the main timer contents. At the next event, the TCx data is transferred to the TCxH register, the TCx is updated and the CxF interrupt flag is set. In all other input capture cases the interrupt flag is set by a valid external event on ICx.<br>0 The timer flags C3F–C0F in TFLG1 are set when a valid input capture transition on the corresponding port pin occurs.<br>1 If in queue mode (BUFEN = 1 and LATQ = 0), the timer flags C3F–C0F in TFLG1 are set only when a latch on the corresponding holding register occurs. If the queue mode is not engaged, the timer flags C3F–C0F are set the same way as for TFMOD = 0. |
| 2<br>PACMX | **8-Bit Pulse Accumulators Maximum Count**<br>0 Normal operation. When the 8-bit pulse accumulator has reached the value 0x00FF, with the next active edge, it will be incremented to 0x0000.<br>1 When the 8-bit pulse accumulator has reached the value 0x00FF, it will not be incremented further. The value 0x00FF indicates a count of 255 or more. |
| 1<br>BUFFEN | **IC Buffer Enable**<br>0 Input capture and pulse accumulator holding registers are disabled.<br>1 Input capture and pulse accumulator holding registers are enabled. The latching mode is defined by LATQ control bit. |
| 0<br>LATQ | **Input Control Latch or Queue Mode Enable** — The BUFEN control bit should be set in order to enable the IC and pulse accumulators holding registers. Otherwise LATQ latching modes are disabled.<br>Write one into ICLAT bit in MCCTL, when LATQ and BUFEN are set will produce latching of input capture and pulse accumulators registers into their holding registers.<br>0 Queue mode of Input Capture is enabled. The main timer value is memorized in the IC register by a valid input pin transition. With a new occurrence of a capture, the value of the IC register will be transferred to its holding register and the IC register memorizes the new timer value.<br>1 Latch mode is enabled. Latching function occurs when modulus down-counter reaches zero or a zero is written into the count register MCCNT (see Section 14.4.1.1.2, “Buffered IC Channels”). With a latching event the contents of IC registers and 8-bit pulse accumulators are transferred to their holding registers. 8-bit pulse accumulators are cleared. |

### 14.3.2.25 Output Compare Pin Disconnect Register (OCPD)

Module Base + 0x002C

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | OCPD7 | OCPD6 | OCPD5 | OCPD4 | OCPD3 | OCPD2 | OCPD1 | OCPD0 |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 14-48. Output Compare Pin Disconnect Register (OCPD)**

Read: Anytime

Write: Anytime

All bits reset to zero.

**Table 14-32. OCPD Field Descriptions**

| Field | Description |
|---|---|
| 7:0<br>OCPD[7:0] | **Output Compare Pin Disconnect Bits**<br>0 Enables the timer channel IO port. Output Compare actions will occur on the channel pin. These bits do not affect the input capture or pulse accumulator functions.<br>1 Disables the timer channel IO port. Output Compare actions will not affect on the channel pin; the output compare flag will still be set on an Output Compare event. |

### 14.3.2.26 Precision Timer Prescaler Select Register (PTPSR)

Module Base + 0x002E

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | PTPS7 | PTPS6 | PTPS5 | PTPS4 | PTPS3 | PTPS2 | PTPS1 | PTPS0 |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 14-49. Precision Timer Prescaler Select Register (PTPSR)**

Read: Anytime

Write: Anytime

All bits reset to zero.

**Table 14-33. PTPSR Field Descriptions**

| Field | Description |
|---|---|
| 7:0<br>PTPS[7:0] | **Precision Timer Prescaler Select Bits** — These eight bits specify the division rate of the main Timer prescaler. These are effective only when the PRNT bit of TSCR1 is set to 1. Table 14-34 shows some selection examples in this case.<br>The newly selected prescale factor will not take effect until the next synchronized edge where all prescale counter stages equal zero. |

**Table 14-34. Precision Timer Prescaler Selection Examples when PRNT = 1**

| PTPS7 | PTPS6 | PTPS5 | PTPS4 | PTPS3 | PTPS2 | PTPS1 | PTPS0 | Prescale Factor |
|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 2 |
| 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 3 |
| 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 4 |
| 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 5 |
| 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 6 |
| 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 7 |
| 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 8 |
| 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 16 |
| 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 32 |
| 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 64 |
| 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 128 |
| 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 256 |

## 14.3.2.27 Precision Timer Modulus Counter Prescaler Select Register (PTMCPSR)

Module Base + 0x002F

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | PTMPS7 | PTMPS6 | PTMPS5 | PTMPS4 | PTMPS3 | PTMPS2 | PTMPS1 | PTMPS0 |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 14-50. Precision Timer Modulus Counter Prescaler Select Register (PTMCPSR)**

Read: Anytime

Write: Anytime

All bits reset to zero.

**Table 14-35. PTMCPSR Field Descriptions**

| Field | Description |
|---|---|
| 7:0<br>PTMPS[7:0] | **Precision Timer Modulus Counter Prescaler Select Bits** — These eight bits specify the division rate of the modulus counter prescaler. These are effective only when the PRNT bit of TSCR1 is set to 1. Table 14-36 shows some possible division rates.<br>The newly selected prescaler division rate will not be effective until a load of the load register into the modulus counter count register occurs. |

**Table 14-36. Precision Timer Modulus Counter Prescaler Select Examples when PRNT = 1**

| PTMPS7 | PTMPS6 | PTMPS5 | PTMPS4 | PTMPS3 | PTMPS2 | PTMPS1 | PTMPS0 | Prescaler Division Rate |
|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 2 |
| 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 3 |
| 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 4 |
| 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 5 |
| 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 6 |
| 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 7 |
| 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 8 |
| 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 16 |
| 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 32 |
| 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 64 |
| 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 128 |
| 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 256 |

## 14.3.2.28 16-Bit Pulse Accumulator B Control Register (PBCTL)

Module Base + 0x0030

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | 0 | PBEN | 0 | 0 | 0 | 0 | PBOVI | 0 |
| W | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented or Reserved

**Figure 14-51. 16-Bit Pulse Accumulator B Control Register (PBCTL)**

Read: Anytime

Write: Anytime

All bits reset to zero.

**Table 14-37. PBCTL Field Descriptions**

| Field | Description |
|---|---|
| 6<br>PBEN | **Pulse Accumulator B System Enable** — PBEN is independent from TEN. With timer disabled, the pulse accumulator can still function unless pulse accumulator is disabled.<br>0 16-bit pulse accumulator system disabled. 8-bit PAC1 and PAC0 can be enabled when their related enable bits in ICPAR are set.<br>1 Pulse accumulator B system enabled. The two 8-bit pulse accumulators PAC1 and PAC0 are cascaded to form the PACB 16-bit pulse accumulator B. When PACB is enabled, the PACN1 and PACN0 registers contents are respectively the high and low byte of the PACB.<br>PA1EN and PA0EN control bits in ICPAR have no effect.<br>The PACB shares the input pin with IC0. |
| 1<br>PBOVI | **Pulse Accumulator B Overflow Interrupt Enable**<br>0 Interrupt inhibited<br>1 Interrupt requested if PBOVF is set |

## 14.3.2.29 Pulse Accumulator B Flag Register (PBFLG)

Module Base + 0x0031

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | 0 | 0 | 0 | 0 | PBOVF | 0 |
| W | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented or Reserved

**Figure 14-52. Pulse Accumulator B Flag Register (PBFLG)**

Read: Anytime

Write used in the flag clearing mechanism. Writing a one to the flag clears the flag. Writing a zero will not affect the current status of the bit.

**NOTE**

When TFFCA = 1, the flag cannot be cleared via the normal flag clearing mechanism (writing a one to the flag). Reference Section 14.3.2.6, “Timer System Control Register 1 (TSCR1)”.

All bits reset to zero.

PBFLG indicates when interrupt conditions have occurred. The flag can be cleared via the normal flag clearing mechanism (writing a one to the flag) or via the fast flag clearing mechanism (Reference TFFCA bit in Section 14.3.2.6, “Timer System Control Register 1 (TSCR1)”).

**Table 14-38. PBFLG Field Descriptions**

| Field | Description |
|---|---|
| 1<br>PBOVF | **Pulse Accumulator B Overflow Flag** — This bit is set when the 16-bit pulse accumulator B overflows from 0xFFFF to 0x0000, or when 8-bit pulse accumulator 1 (PAC1) overflows from 0x00FF to 0x0000.<br>When PACMX = 1, PBOVF bit can also be set if 8-bit pulse accumulator 1 (PAC1) reaches 0x00FF and an active edge follows on IC1. |

### 14.3.2.30 8-Bit Pulse Accumulators Holding Registers (PA3H–PA0H)

Module Base + 0x0032

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | PA3H7 | PA3H6 | PA3H5 | PA3H4 | PA3H3 | PA3H2 | PA3H1 | PA3H0 |
| W | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented or Reserved

**Figure 14-53. 8-Bit Pulse Accumulators Holding Register 3 (PA3H)**

Module Base + 0x0033

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | PA2H7 | PA2H6 | PA2H5 | PA2H4 | PA2H3 | PA2H2 | PA2H1 | PA2H0 |
| W | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented or Reserved

**Figure 14-54. 8-Bit Pulse Accumulators Holding Register 2 (PA2H)**

Module Base + 0x0034

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | PA1H7 | PA1H6 | PA1H5 | PA1H4 | PA1H3 | PA1H2 | PA1H1 | PA1H0 |
| W | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented or Reserved

**Figure 14-55. 8-Bit Pulse Accumulators Holding Register 1 (PA1H)**

Module Base + 0x0035

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | PA0H7 | PA0H6 | PA0H5 | PA0H4 | PA0H3 | PA0H2 | PA0H1 | PA0H0 |
| W | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented or Reserved

**Figure 14-56. 8-Bit Pulse Accumulators Holding Register 0 (PA0H)**

Read: Anytime.

Write: Has no effect.

All bits reset to zero.

These registers are used to latch the value of the corresponding pulse accumulator when the related bits in register ICPAR are enabled (see Section 14.4.1.3, “Pulse Accumulators”).

### 14.3.2.31 Modulus Down-Counter Count Register (MCCNT)

Module Base + 0x0036

| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | MCCNT15 | MCCNT14 | MCCNT13 | MCCNT12 | MCCNT11 | MCCNT10 | MCCNT9 | MCCNT8 |
| Reset | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

**Figure 14-57. Modulus Down-Counter Count Register High (MCCNT)**

Module Base + 0x0037

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | MCCNT7 | MCCNT6 | MCCNT5 | MCCNT4 | MCCNT3 | MCCNT2 | MCCNT1 | MCCNT0 |
| Reset | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

**Figure 14-58. Modulus Down-Counter Count Register Low (MCCNT)**

Read: Anytime

Write: Anytime.

All bits reset to one.

A full access for the counter register will take place in one clock cycle.

**NOTE**

A separate read/write for high byte and low byte will give different results than accessing them as a word.

If the RDMCL bit in MCCTL register is cleared, reads of the MCCNT register will return the present value of the count register. If the RDMCL bit is set, reads of the MCCNT will return the contents of the load register.

If a 0x0000 is written into MCCNT when LATQ and BUFEN in ICSYS register are set, the input capture and pulse accumulator registers will be latched.

With a 0x0000 write to the MCCNT, the modulus counter will stay at zero and does not set the MCZF flag in MCFLG register.

If the modulus down counter is enabled (MCEN = 1) and modulus mode is enabled (MODMC = 1), a write to MCCNT will update the load register with the value written to it. The count register will not be updated with the new value until the next counter underflow.

If modulus mode is not enabled (MODMC = 0), a write to MCCNT will clear the modulus prescaler and will immediately update the counter register with the value written to it and down-counts to 0x0000 and stops.

The FLMC bit in MCCTL can be used to immediately update the count register with the new value if an immediate load is desired.

### 14.3.2.32 Timer Input Capture Holding Registers 0–3 (TCxH)

Module Base + 0x0038

| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|---|
| R | TC15 | TC14 | TC13 | TC12 | TC11 | TC10 | TC9 | TC8 |
| W | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented or Reserved

**Figure 14-59. Timer Input Capture Holding Register 0 High (TC0H)**

Module Base + 0x0039

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | TC7 | TC6 | TC5 | TC4 | TC3 | TC2 | TC1 | TC0 |
| W | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented or Reserved

**Figure 14-60. Timer Input Capture Holding Register 0 Low (TC0H)**

Module Base + 0x003A

| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|---|
| R | TC15 | TC14 | TC13 | TC12 | TC11 | TC10 | TC9 | TC8 |
| W | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented or Reserved

**Figure 14-61. Timer Input Capture Holding Register 1 High (TC1H)**

Module Base + 0x003B

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | TC7 | TC6 | TC5 | TC4 | TC3 | TC2 | TC1 | TC0 |
| W | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented or Reserved

**Figure 14-62. Timer Input Capture Holding Register 1 Low (TC1H)**

Module Base + 0x003C

| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|---|
| R | TC15 | TC14 | TC13 | TC12 | TC11 | TC10 | TC9 | TC8 |
| W | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented or Reserved

**Figure 14-63. Timer Input Capture Holding Register 2 High (TC2H)**

Module Base + 0x003D

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | TC7 | TC6 | TC5 | TC4 | TC3 | TC2 | TC1 | TC0 |
| W | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented or Reserved

**Figure 14-64. Timer Input Capture Holding Register 2 Low (TC2H)**

Module Base + 0x003E

| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|---|
| R | TC15 | TC14 | TC13 | TC12 | TC11 | TC10 | TC9 | TC8 |
| W | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented or Reserved

**Figure 14-65. Timer Input Capture Holding Register 3 High (TC3H)**

Module Base + 0x003F

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | TC7 | TC6 | TC5 | TC4 | TC3 | TC2 | TC1 | TC0 |
| W | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented or Reserved

**Figure 14-66. Timer Input Capture Holding Register 3 Low (TC3H)**

Read: Anytime

Write: Has no effect.

All bits reset to zero.

These registers are used to latch the value of the input capture registers TC0–TC3. The corresponding IOSx bits in TIOS should be cleared (see Section 14.4.1.1, “IC Channels”).

## 14.4 Functional Description

This section provides a complete functional description of the ECT block, detailing the operation of the design from the end user perspective in a number of subsections.

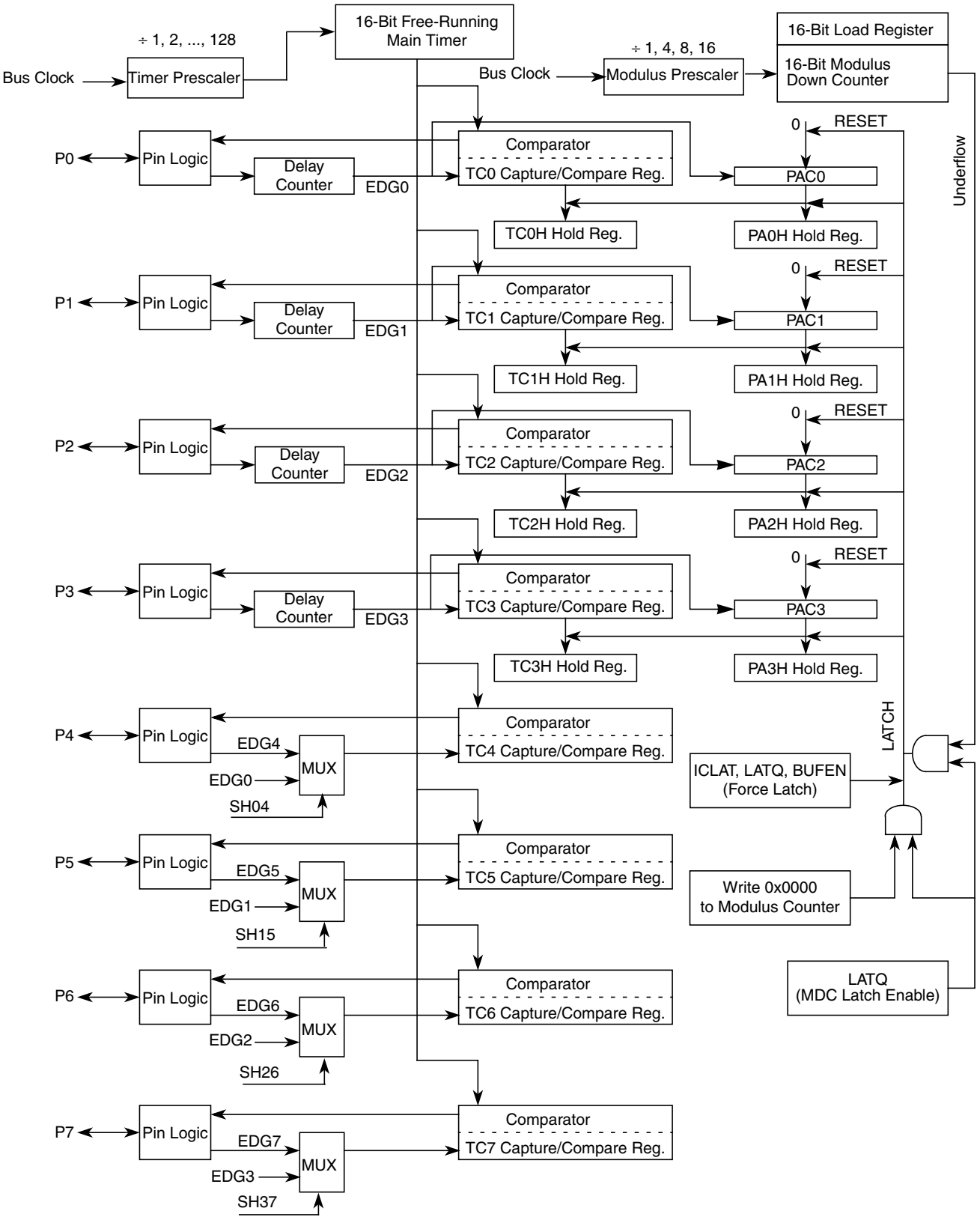



**Figure 14-67. Detailed Timer Block Diagram in Latch Mode when PRNT = 0**

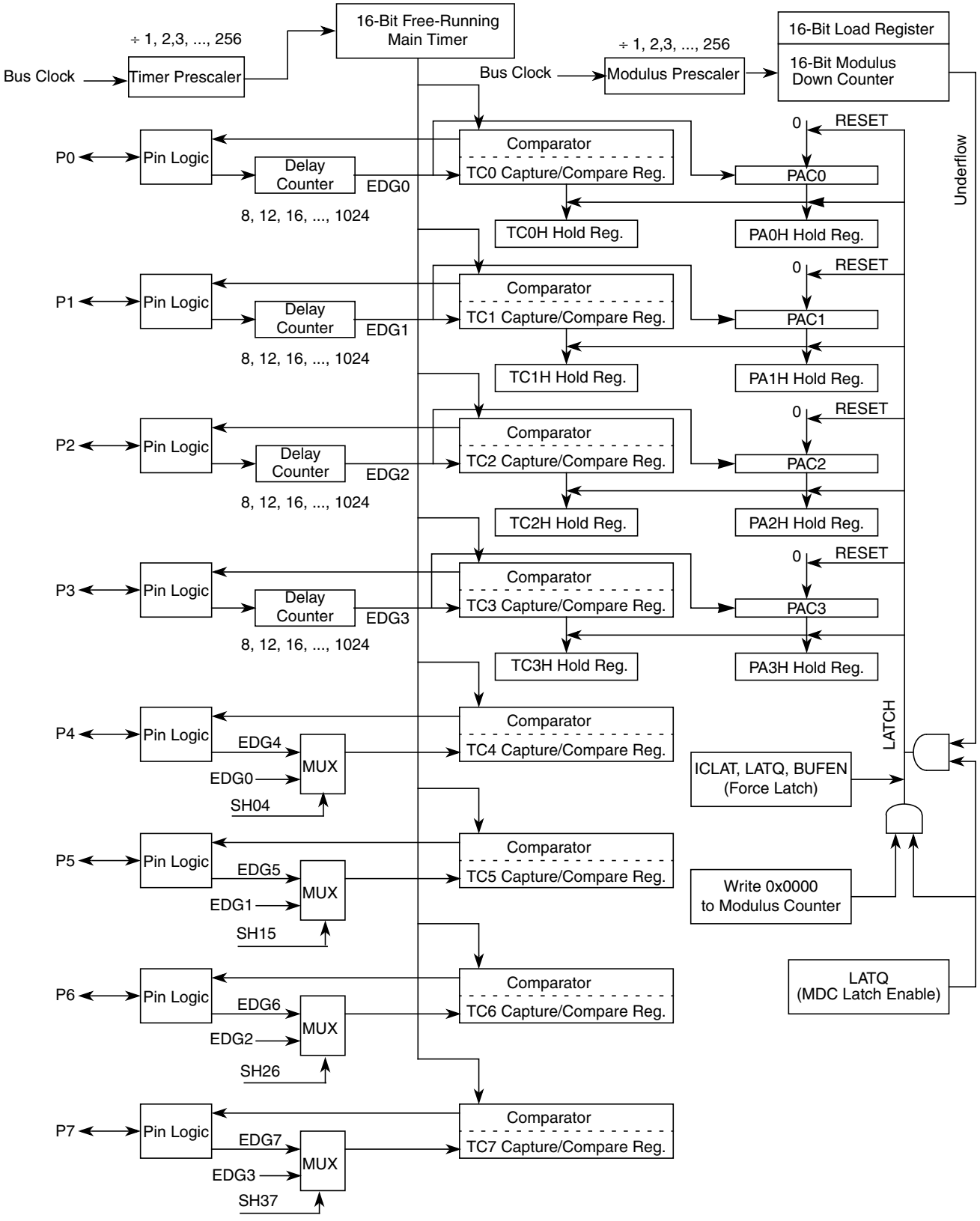



**Figure 14-68. Detailed Timer Block Diagram in Latch Mode when PRNT = 1**

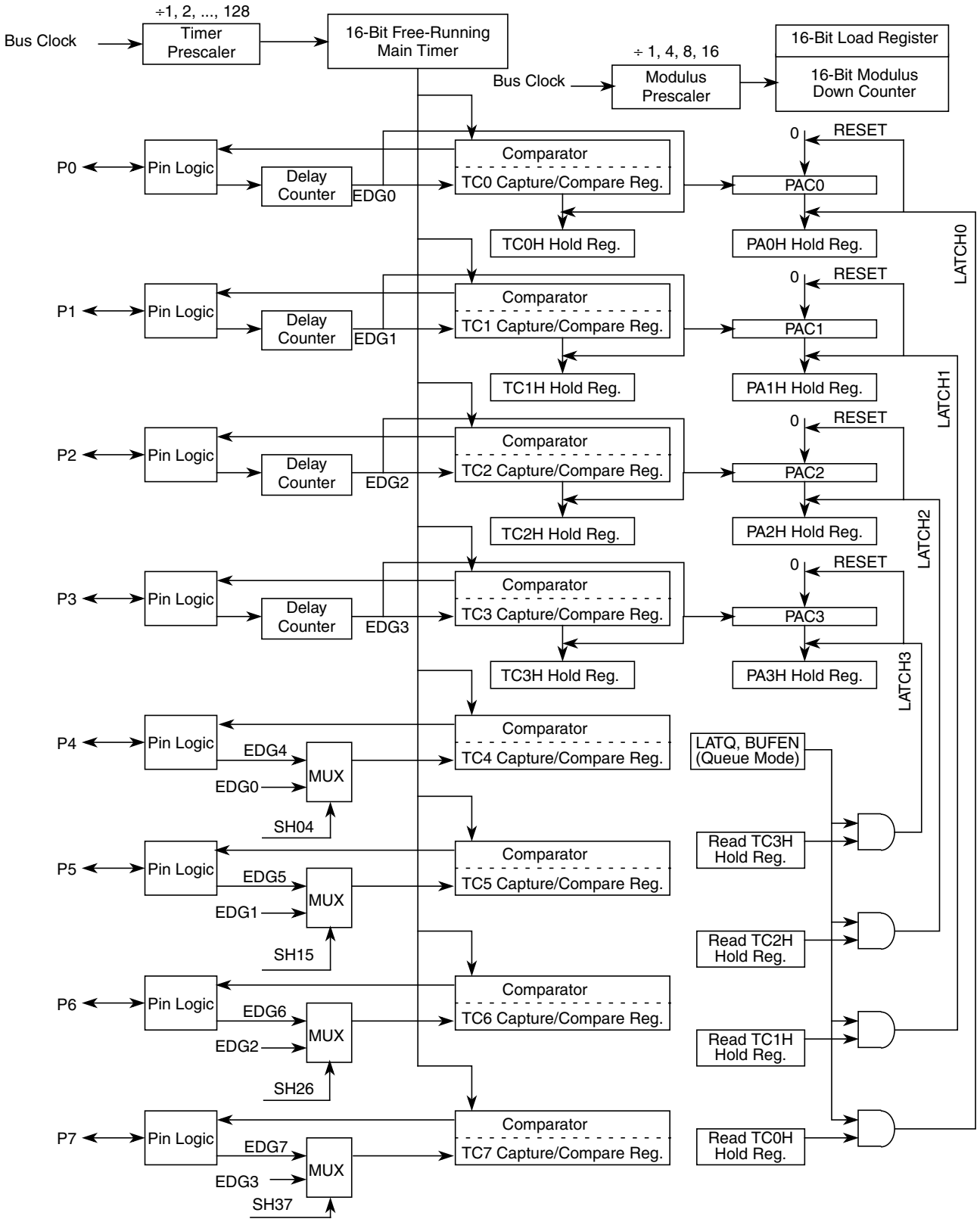



**Figure 14-69. Detailed Timer Block Diagram in Queue Mode when PRNT = 0**

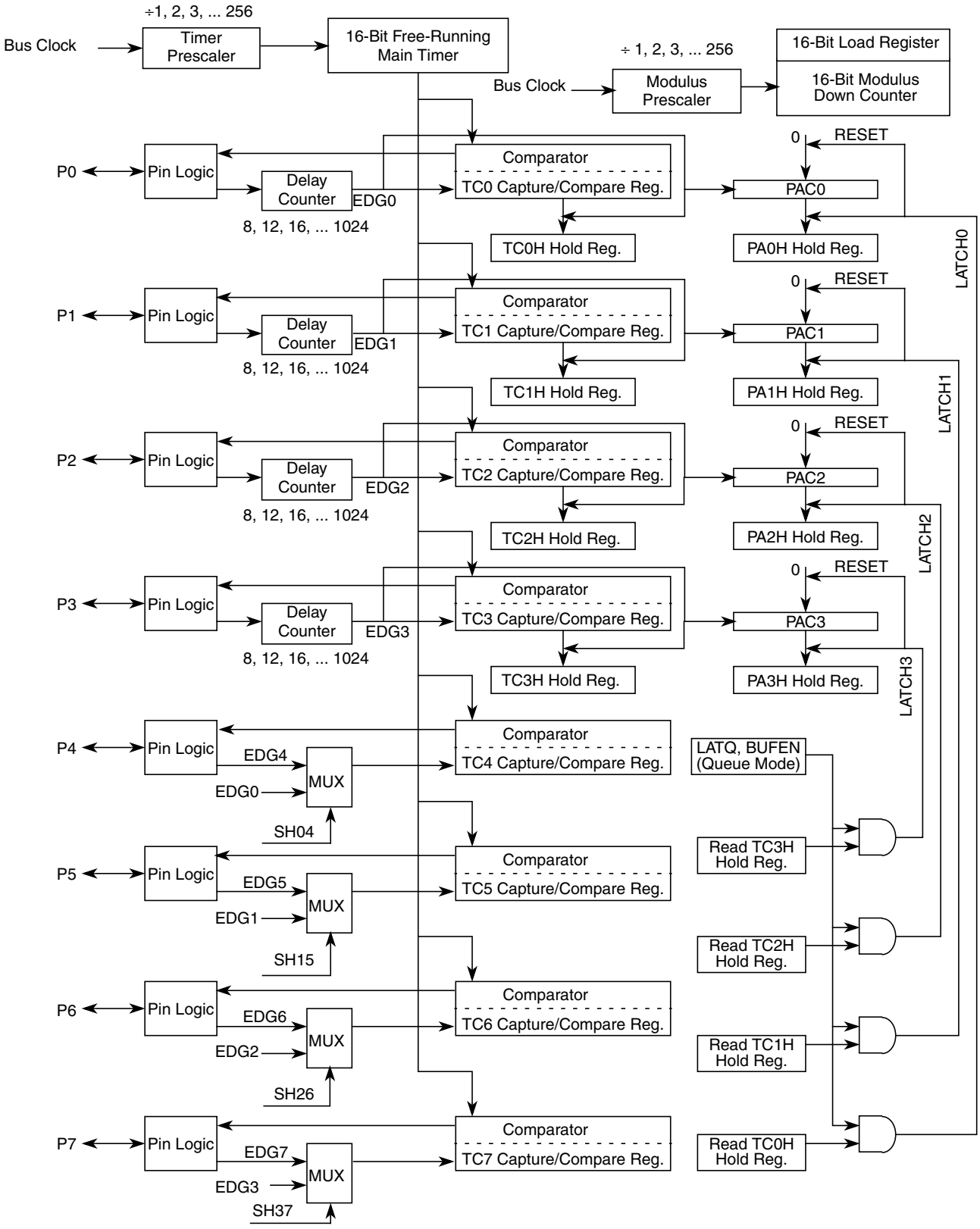



**Figure 14-70. Detailed Timer Block Diagram in Queue Mode when PRNT = 1**

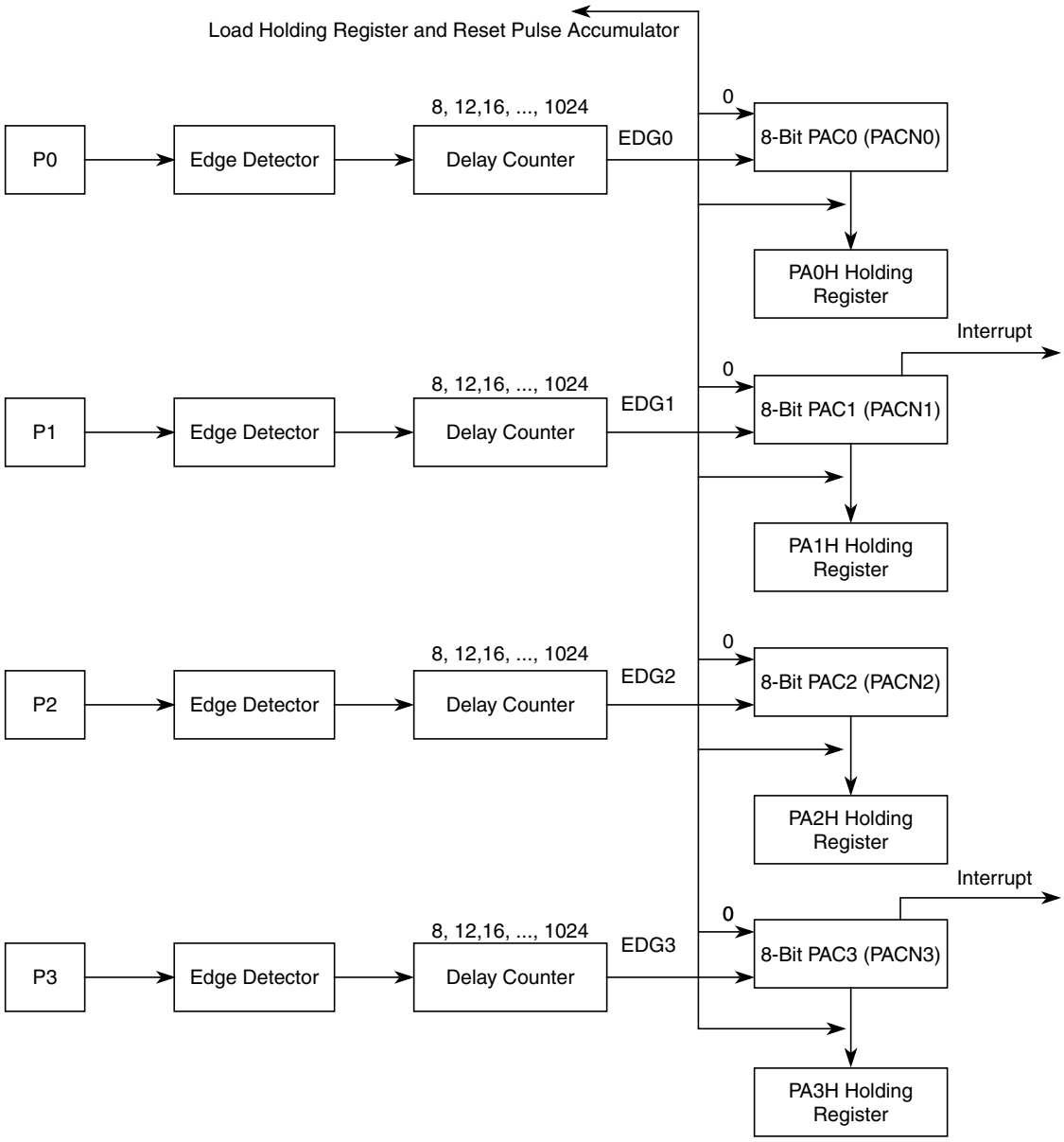



**Figure 14-71. 8-Bit Pulse Accumulators Block Diagram**

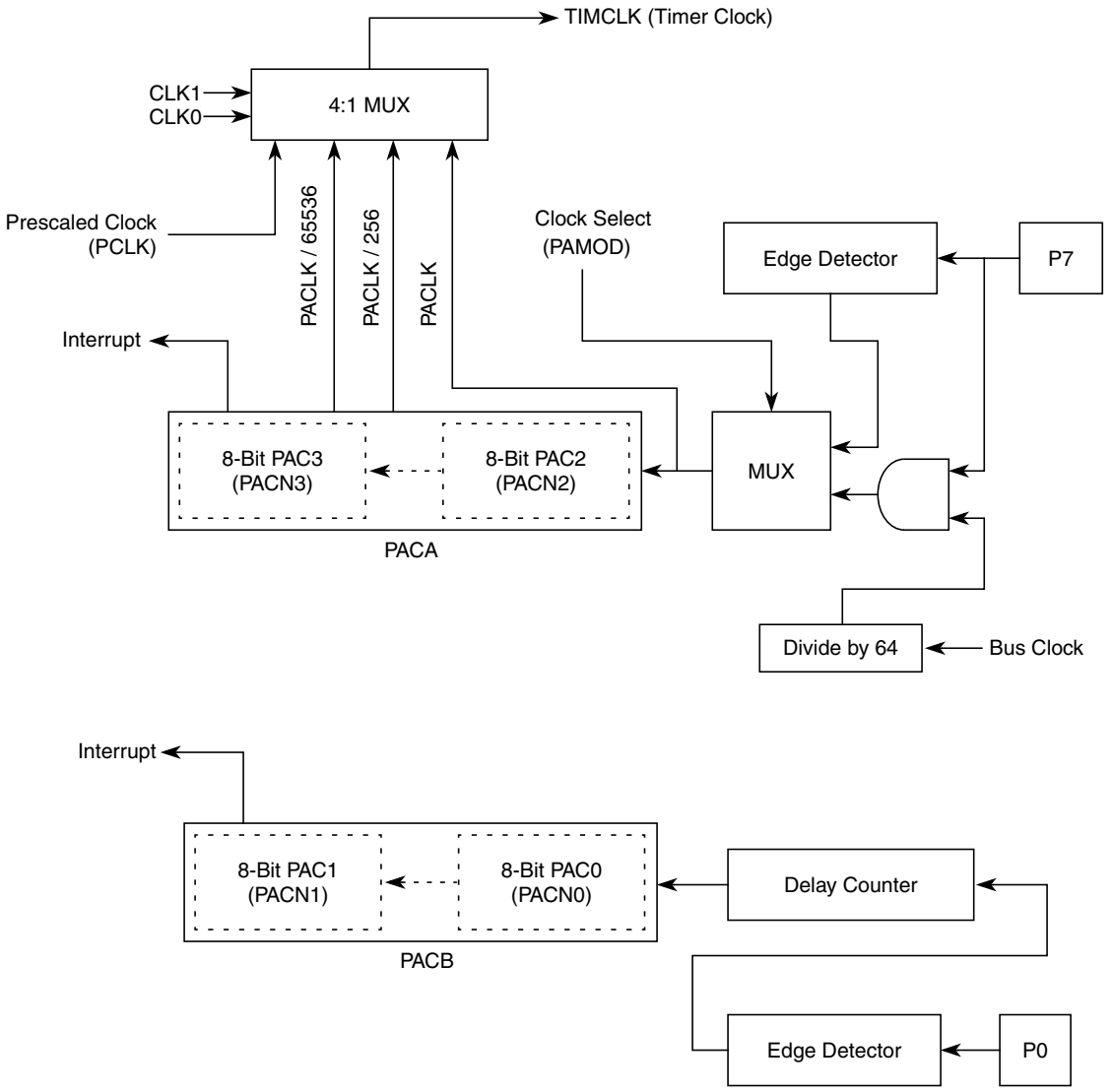



**Figure 14-72. 16-Bit Pulse Accumulators Block Diagram**

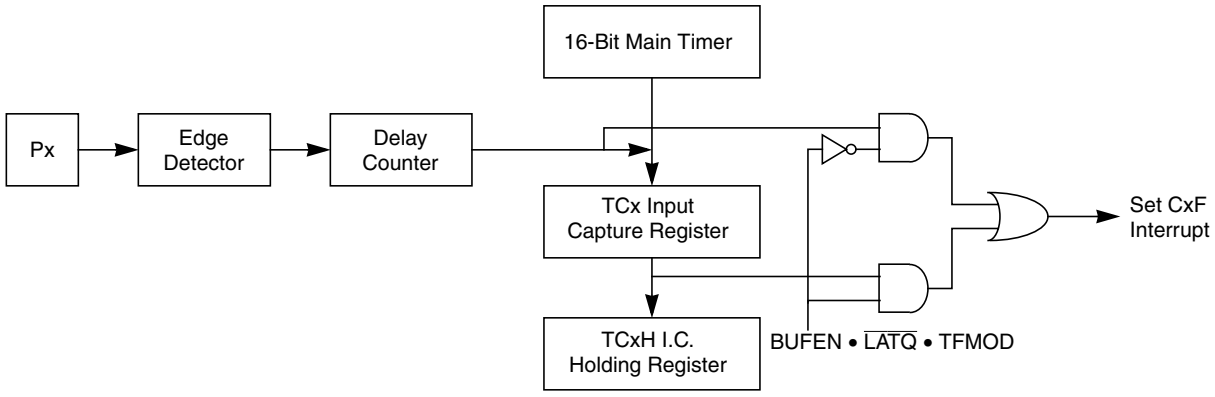



**Figure 14-73. Block Diagram for Port 7 with Output Compare/Pulse Accumulator A**

### 14.4.1 Enhanced Capture Timer Modes of Operation

The enhanced capture timer has 8 input capture, output compare (IC/OC) channels, same as on the HC12 standard timer (timer channels TC0 to TC7). When channels are selected as input capture by selecting the IOSx bit in TIOS register, they are called input capture (IC) channels.

Four IC channels (channels 7–4) are the same as on the standard timer with one capture register each that memorizes the timer value captured by an action on the associated input pin.

Four other IC channels (channels 3–0), in addition to the capture register, also have one buffer each called a holding register. This allows two different timer values to be saved without generating any interrupts.

Four 8-bit pulse accumulators are associated with the four buffered IC channels (channels 3–0). Each pulse accumulator has a holding register to memorize their value by an action on its external input. Each pair of pulse accumulators can be used as a 16-bit pulse accumulator.

The 16-bit modulus down-counter can control the transfer of the IC registers and the pulse accumulators contents to the respective holding registers for a given period, every time the count reaches zero.

The modulus down-counter can also be used as a stand-alone time base with periodic interrupt capability.

#### 14.4.1.1 IC Channels

The IC channels are composed of four standard IC registers and four buffered IC channels.

- An IC register is empty when it has been read or latched into the holding register.
- A holding register is empty when it has been read.

##### 14.4.1.1.1 Non-Buffered IC Channels

The main timer value is memorized in the IC register by a valid input pin transition. If the corresponding NOVWx bit of the ICOVW register is cleared, with a new occurrence of a capture, the contents of IC register are overwritten by the new value. If the corresponding NOVWx bit of the ICOVW register is set, the capture register cannot be written unless it is empty. This will prevent the captured value from being overwritten until it is read.

##### 14.4.1.1.2 Buffered IC Channels

There are two modes of operations for the buffered IC channels:

1. IC latch mode (LATQ = 1)

   The main timer value is memorized in the IC register by a valid input pin transition (see Figure 14-67 and Figure 14-68).

   The value of the buffered IC register is latched to its holding register by the modulus counter for a given period when the count reaches zero, by a write 0x0000 to the modulus counter or by a write to ICLAT in the MCCTL register.

   If the corresponding NOVWx bit of the ICOVW register is cleared, with a new occurrence of a capture, the contents of IC register are overwritten by the new value. In case of latching, the contents of its holding register are overwritten.

If the corresponding NOVWx bit of the ICOVW register is set, the capture register or its holding register cannot be written by an event unless they are empty (see Section 14.4.1.1, “IC Channels”). This will prevent the captured value from being overwritten until it is read or latched in the holding register.

2. IC Queue Mode (LATQ = 0)

   The main timer value is memorized in the IC register by a valid input pin transition (see Figure 14-69 and Figure 14-70).

   If the corresponding NOVWx bit of the ICOVW register is cleared, with a new occurrence of a capture, the value of the IC register will be transferred to its holding register and the IC register memorizes the new timer value.

   If the corresponding NOVWx bit of the ICOVW register is set, the capture register or its holding register cannot be written by an event unless they are empty (see Section 14.4.1.1, “IC Channels”).

   if the TFMOD bit of the ICSYS register is set,the timer flags C3F--C0F in TFLG register are set only when a latch on the corresponding holding register occurs,after C3F--C0F are set,user should clear flag C3F--C0F,then read TCx and TCxH to make TCx and TCxH be empty.

   In queue mode, reads of the holding register will latch the corresponding pulse accumulator value to its holding register.

#### 14.4.1.1.3 Delayed IC Channels

There are four delay counters in this module associated with IC channels 0–3. The use of this feature is explained in the diagram and notes below.

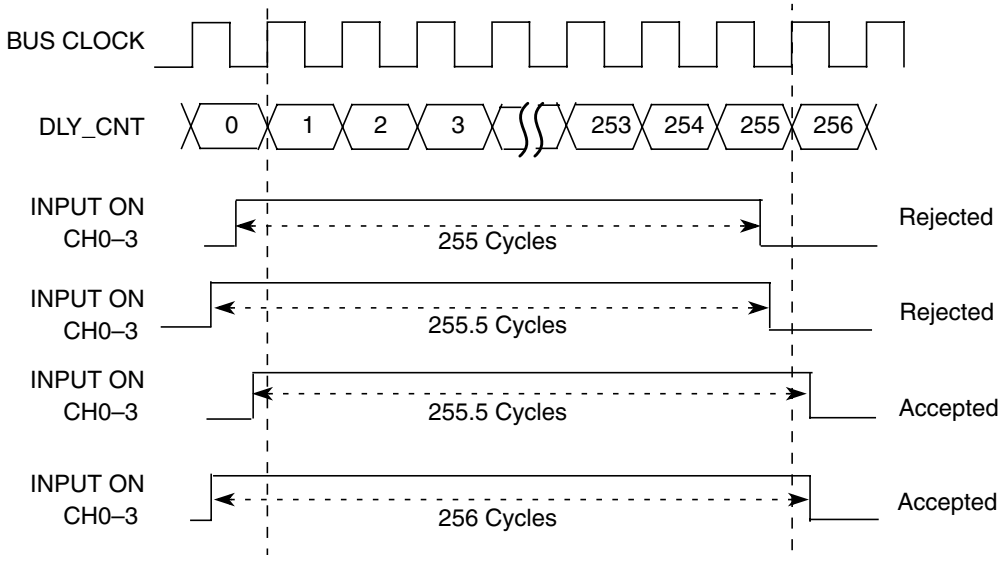



**Figure 14-74. Channel Input Validity with Delay Counter Feature**

In Figure 14-74 a delay counter value of 256 bus cycles is considered.

1. Input pulses with a duration of (DLY_CNT – 1) cycles or shorter are rejected.
2. Input pulses with a duration between (DLY_CNT – 1) and DLY_CNT cycles may be rejected or accepted, depending on their relative alignment with the sample points.

3. Input pulses with a duration between (DLY_CNT – 1) and DLY_CNT cycles may be rejected or accepted, depending on their relative alignment with the sample points.
4. Input pulses with a duration of DLY_CNT or longer are accepted.

#### 14.4.1.2 OC Channel Initialization

An internal compare channel whose output drives OCx may be programmed before the timer drives the output compare state (OCx). The required output of the compare logic can be disconnected from the pin, leaving it driven by the GP IO port, by setting the appropriate OCPDx bit before enabling the output compare channel (by default the OCPD bits are cleared which would enable the output compare logic to drive the pin as soon as the timer output compare channel is enabled). The desired initial state can then be configured in the internal output compare logic by forcing a compare action with the logic disconnected from the IO (by writing a one to CFORCx bit with TIOSx, OCPDx and TEN bits set to one). Clearing the output compare disconnect bit (OCPDx) will then allow the internal compare logic to drive the programmed state to OCx. This allows a glitch free switching between general purpose I/O and timer output functionality.

#### 14.4.1.3 Pulse Accumulators

There are four 8-bit pulse accumulators with four 8-bit holding registers associated with the four IC buffered channels 3–0. A pulse accumulator counts the number of active edges at the input of its channel.

The minimum pulse width for the PAI input is greater than two bus clocks.The maximum input frequency on the pulse accumulator channel is one half the bus frequency or Eclk.

The user can prevent the 8-bit pulse accumulators from counting further than 0x00FF by utilizing the PACMX control bit in the ICSYS register. In this case, a value of 0x00FF means that 255 counts or more have occurred.

Each pair of pulse accumulators can be used as a 16-bit pulse accumulator (see Figure 14-72).

Pulse accumulator B operates only as an event counter, it does not feature gated time accumulation mode. The edge control for pulse accumulator B as a 16-bit pulse accumulator is defined by TCTL4[1:0].

To operate the 16-bit pulse accumulators A and B (PACA and PACB) independently of input capture or output compare 7 and 0 respectively, the user must set the corresponding bits: IOSx = 1, OMx = 0, and OLx = 0. OC7M7 or OC7M0 in the OC7M register must also be cleared.

There are two modes of operation for the pulse accumulators:

- Pulse accumulator latch mode

  The value of the pulse accumulator is transferred to its holding register when the modulus down-counter reaches zero, a write 0x0000 to the modulus counter or when the force latch control bit ICLAT is written.

  At the same time the pulse accumulator is cleared.
- Pulse accumulator queue mode

  When queue mode is enabled, reads of an input capture holding register will transfer the contents of the associated pulse accumulator to its holding register.

At the same time the pulse accumulator is cleared.

#### 14.4.1.4 Modulus Down-Counter

The modulus down-counter can be used as a time base to generate a periodic interrupt. It can also be used to latch the values of the IC registers and the pulse accumulators to their holding registers.

The action of latching can be programmed to be periodic or only once.

#### 14.4.1.5 Precision Timer

By enabling the PRNT bit of the TSCR1 register, the performance of the timer can be enhanced. In this case, it is possible to set additional prescaler settings for the main timer counter and modulus down counter and enhance delay counter settings compared to the settings in the present ECT timer.

#### 14.4.1.6 Flag Clearing Mechanisms

The flags in the ECT can be cleared one of two ways:

1. Normal flag clearing mechanism (TFFCA = 0)

   Any of the ECT flags can be cleared by writing a one to the flag.
2. Fast flag clearing mechanism (TFFCA = 1)

   With the timer fast flag clear all (TFFCA) enabled, the ECT flags can only be cleared by accessing the various registers associated with the ECT modes of operation as described below. The flags cannot be cleared via the normal flag clearing mechanism. This fast flag clearing mechanism has the advantage of eliminating the software overhead required by a separate clear sequence. Extra care must be taken to avoid accidental flag clearing due to unintended accesses.

   — Input capture

   A read from an input capture channel register causes the corresponding channel flag, CxF, to be cleared in the TFLG1 register.

   — Output compare

   A write to the output compare channel register causes the corresponding channel flag, CxF, to be cleared in the TFLG1 register.

   — Timer counter

   Any access to the TCNT register clears the TOF flag in the TFLG2 register.

   — Pulse accumulator A

   Any access to the PACN3 and PACN2 registers clears the PAOVF and PAIF flags in the PAFLG register.

   — Pulse accumulator B

   Any access to the PACN1 and PACN0 registers clears the PBOVF flag in the PBFLG register.

   — Modulus down counter

   Any access to the MCCNT register clears the MCZF flag in the MCFLG register.

### 14.4.2 Reset

The reset state of each individual bit is listed within the register description section (Section 14.3, “Memory Map and Register Definition”) which details the registers and their bit-fields.

### 14.4.3 Interrupts

This section describes interrupts originated by the ECT block. The MCU must service the interrupt requests. Table 14-39 lists the interrupts generated by the ECT to communicate with the MCU.

**Table 14-39. ECT Interrupts**

| Interrupt Source | Description |
|---|---|
| Timer channel 7–0 | Active high timer channel interrupts 7–0 |
| Modulus counter underflow | Active high modulus counter interrupt |
| Pulse accumulator B overflow | Active high pulse accumulator B interrupt |
| Pulse accumulator A input | Active high pulse accumulator A input interrupt |
| Pulse accumulator A overflow | Pulse accumulator overflow interrupt |
| Timer overflow | Timer 0verflow interrupt |

The ECT only originates interrupt requests. The following is a description of how the module makes a request and how the MCU should acknowledge that request. The interrupt vector offset and interrupt number are chip dependent.

#### 14.4.3.1 Channel [7:0] Interrupt

This active high output will be asserted by the module to request a timer channel 7–0 interrupt to be serviced by the system controller.

#### 14.4.3.2 Modulus Counter Interrupt

This active high output will be asserted by the module to request a modulus counter underflow interrupt to be serviced by the system controller.

#### 14.4.3.3 Pulse Accumulator B Overflow Interrupt

This active high output will be asserted by the module to request a timer pulse accumulator B overflow interrupt to be serviced by the system controller.

#### 14.4.3.4 Pulse Accumulator A Input Interrupt

This active high output will be asserted by the module to request a timer pulse accumulator A input interrupt to be serviced by the system controller.

#### 14.4.3.5 Pulse Accumulator A Overflow Interrupt

This active high output will be asserted by the module to request a timer pulse accumulator A overflow interrupt to be serviced by the system controller.

#### 14.4.3.6 Timer Overflow Interrupt

This active high output will be asserted by the module to request a timer overflow interrupt to be serviced by the system controller.

# Chapter 15
# Inter-Integrated Circuit (IICV3) Block Description

**Table 15-1. Revision History**

| Revision Number | Revision Date | Sections Affected | Description of Changes |
|---|---|---|---|
| V01.03 | 28 Jul 2006 | 15.7.1.7/15-601 | - Update flow-chart of interrupt routine for 10-bit address |
| V01.04 | 17 Nov 2006 | 15.3.1.2/15-582 | - Revise Table1-5 |
| V01.05 | 14 Aug 2007 | 15.3.1.1/15-581 | - Backward compatible for IBAD bit name |

## 15.1 Introduction

The inter-IC bus (IIC) is a two-wire, bidirectional serial bus that provides a simple, efficient method of data exchange between devices. Being a two-wire device, the IIC bus minimizes the need for large numbers of connections between devices, and eliminates the need for an address decoder.

This bus is suitable for applications requiring occasional communications over a short distance between a number of devices. It also provides flexibility, allowing additional devices to be connected to the bus for further expansion and system development.

The interface is designed to operate up to 100 kbps with maximum bus loading and timing. The device is capable of operating at higher baud rates, up to a maximum of clock/20, with reduced bus loading. The maximum communication length and the number of devices that can be connected are limited by a maximum bus capacitance of 400 pF.

### 15.1.1 Features

The IIC module has the following key features:

- Compatible with I2C bus standard
- Multi-master operation
- Software programmable for one of 256 different serial clock frequencies
- Software selectable acknowledge bit
- Interrupt driven byte-by-byte data transfer
- Arbitration lost interrupt with automatic mode switching from master to slave
- Calling address identification interrupt
- Start and stop signal generation/detection
- Repeated start signal generation
- Acknowledge bit generation/detection
- Bus busy detection

- General Call Address detection
- Compliant to ten-bit address

### 15.1.2 Modes of Operation

The IIC functions the same in normal, special, and emulation modes. It has two low power modes: wait and stop modes.

### 15.1.3 Block Diagram

The block diagram of the IIC module is shown in Figure 15-1.

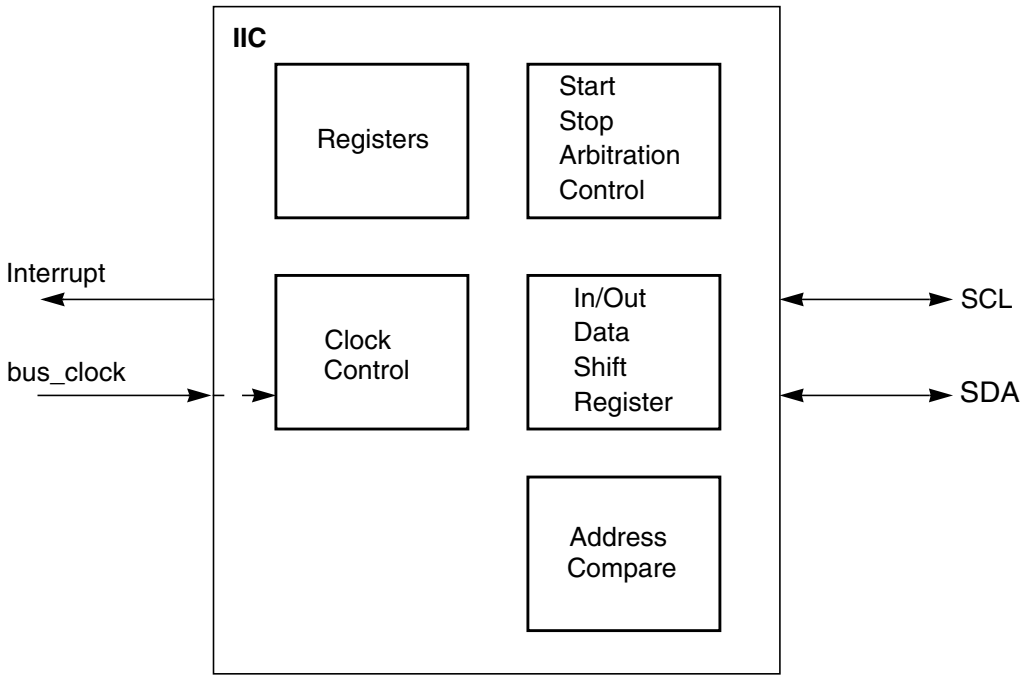



**Figure 15-1. IIC Block Diagram**

## 15.2 External Signal Description

The IICV3 module has two external pins.

### 15.2.1 IIC_SCL — Serial Clock Line Pin

This is the bidirectional serial clock line (SCL) of the module, compatible to the IIC bus specification.

### 15.2.2 IIC_SDA — Serial Data Line Pin

This is the bidirectional serial data line (SDA) of the module, compatible to the IIC bus specification.

## 15.3 Memory Map and Register Definition

This section provides a detailed description of all memory and registers for the IIC module.

### 15.3.1 Register Descriptions

This section consists of register descriptions in address order. Each description includes a standard register diagram with an associated figure number. Details of register bit and field function follow the register diagrams, in bit order.

| Register Name | | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|---|
| 0x0000 IBAD | R | ADR7 | ADR6 | ADR5 | ADR4 | ADR3 | ADR2 | ADR1 | 0 |
| | W | | | | | | | | |
| 0x0001 IBFD | R | IBC7 | IBC6 | IBC5 | IBC4 | IBC3 | IBC2 | IBC1 | IBC0 |
| | W | | | | | | | | |
| 0x0002 IBCR | R | IBEN | IBIE | MS/$\overline{SL}$ | Tx/$\overline{Rx}$ | TXAK | 0 | 0 | IBSWAI |
| | W | | | | | | RSTA | | |
| 0x0003 IBSR | R | TCF | IAAS | IBB | IBAL | 0 | SRW | IBIF | RXAK |
| | W | | | | | | | | |
| 0x0004 IBDR | R | D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 |
| | W | | | | | | | | |
| 0x0005 IBCR2 | R | GCEN | ADTYPE | 0 | 0 | 0 | ADR10 | ADR9 | ADR8 |
| | W | | | | | | | | |

= Unimplemented or Reserved

**Figure 15-2. IIC Register Summary**

#### 15.3.1.1 IIC Address Register (IBAD)

Module Base +0x0000

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | ADR7 | ADR6 | ADR5 | ADR4 | ADR3 | ADR2 | ADR1 | 0 |
| W | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented or Reserved

**Figure 15-3. IIC Bus Address Register (IBAD)**

Read and write anytime

This register contains the address the IIC bus will respond to when addressed as a slave; note that it is not the address sent on the bus during the address transfer.

**Table 15-2. IBAD Field Descriptions**

| Field | Description |
|---|---|
| 7:1<br>ADR[7:1] | **Slave Address** — Bit 1 to bit 7 contain the specific slave address to be used by the IIC bus module.The default mode of IIC bus is slave mode for an address match on the bus. |
| 0<br>Reserved | Reserved — Bit 0 of the IBAD is reserved for future compatibility. This bit will always read 0. |

### 15.3.1.2 IIC Frequency Divider Register (IBFD)

Module Base + 0x0001

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | IBC7 | IBC6 | IBC5 | IBC4 | IBC3 | IBC2 | IBC1 | IBC0 |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented or Reserved

**Figure 15-4. IIC Bus Frequency Divider Register (IBFD)**

Read and write anytime

**Table 15-3. IBFD Field Descriptions**

| Field | Description |
|---|---|
| 7:0<br>IBC[7:0] | **I Bus Clock Rate 7:0** — This field is used to prescale the clock for bit rate selection. The bit clock generator is implemented as a prescale divider — IBC7:6, prescaled shift register — IBC5:3 select the prescaler divider and IBC2-0 select the shift register tap point. The IBC bits are decoded to give the tap and prescale values as shown in Table 15-4. |

**Table 15-4. I-Bus Tap and Prescale Values**

| IBC2-0 (bin) | SCL Tap (clocks) | SDA Tap (clocks) |
|---|---|---|
| 000 | 5 | 1 |
| 001 | 6 | 1 |
| 010 | 7 | 2 |
| 011 | 8 | 2 |
| 100 | 9 | 3 |
| 101 | 10 | 3 |
| 110 | 12 | 4 |
| 111 | 15 | 4 |

**Table 15-5. Prescale Divider Encoding**

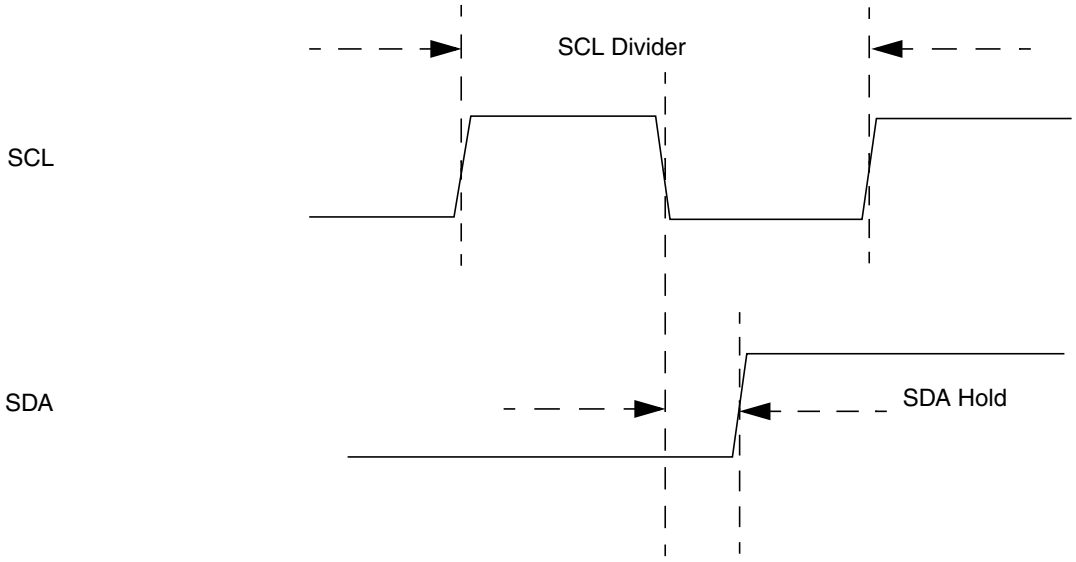
| IBC5-3 (bin) | scl2start (clocks) | scl2stop (clocks) | scl2tap (clocks) | tap2tap (clocks) |
|---|---|---|---|---|
| 000 | 2 | 7 | 4 | 1 |
| 001 | 2 | 7 | 4 | 2 |
| 010 | 2 | 9 | 6 | 4 |
| 011 | 6 | 9 | 6 | 8 |
| 100 | 14 | 17 | 14 | 16 |
| 101 | 30 | 33 | 30 | 32 |
| 110 | 62 | 65 | 62 | 64 |
| 111 | 126 | 129 | 126 | 128 |

**Table 15-6. Multiplier Factor**

| IBC7-6 | MUL |
|---|---|
| 00 | 01 |
| 01 | 02 |
| 10 | 04 |
| 11 | RESERVED |

The number of clocks from the falling edge of SCL to the first tap (Tap[1]) is defined by the values shown in the scl2tap column of Table 15-4, all subsequent tap points are separated by $2^{IBC5-3}$ as shown in the tap2tap column in Table 15-4. The SCL Tap is used to generated the SCL period and the SDA Tap is used to determine the delay from the falling edge of SCL to SDA changing, the SDA hold time.

IBC7–6 defines the multiplier factor MUL. The values of MUL are shown in the Table 15-6.



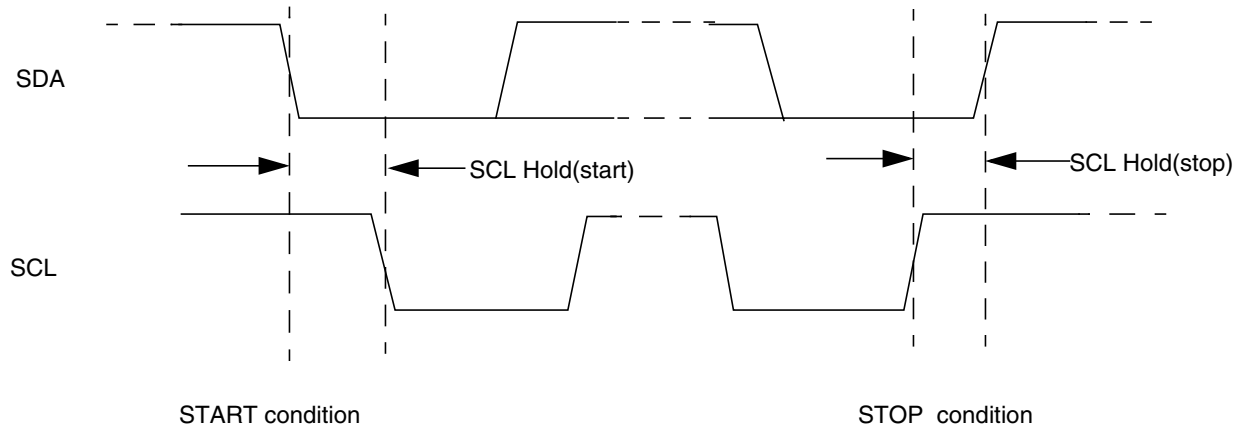



**Figure 15-5. SCL Divider and SDA Hold**

The equation used to generate the divider values from the IBFD bits is:

**SCL Divider = MUL x {2 x (scl2tap + [(SCL_Tap -1) x tap2tap] + 2)}**

The SDA hold delay is equal to the CPU clock period multiplied by the SDA Hold value shown in Table 15-7. The equation used to generate the SDA Hold value from the IBFD bits is:

**SDA Hold = MUL x {scl2tap + [(SDA_Tap - 1) x tap2tap] + 3}**

The equation for SCL Hold values to generate the start and stop conditions from the IBFD bits is:

**SCL Hold(start) = MUL x [scl2start + (SCL_Tap - 1) x tap2tap]**

**SCL Hold(stop) = MUL x [scl2stop + (SCL_Tap - 1) x tap2tap]**

**NOTE**

A master SCL divider period can be prolonged at higher internal bus frequencies. This happens when the internal bus cycle length becomes equal to a pad delay. The SCL input is used for clock arbitration of multiple masters. Thus after each SCL edge is internally driven an extra bus period is counted before the pad level is attained, allowing the next toggle. This has the effect of extending the SCL Divider values in Table 15-7 for MUL=1 and IBC[7:0] = 0x00 to 0x0F.

**Table 15-7. IIC Divider and Hold Values (Sheet 1 of 6)**

| IBC[7:0] (hex) | SCL Divider (clocks) | SDA Hold (clocks) | SCL Hold (start) | SCL Hold (stop) |
|---|---|---|---|---|
| **MUL=1** | | | | |
| 00 | 20 | 7 | 6 | 11 |
| 01 | 22 | 7 | 7 | 12 |
| 02 | 24 | 8 | 8 | 13 |
| 03 | 26 | 8 | 9 | 14 |
| 04 | 28 | 9 | 10 | 15 |
| 05 | 30 | 9 | 11 | 16 |

**Table 15-7. IIC Divider and Hold Values (Sheet 2 of 6)**

| IBC[7:0] (hex) | SCL Divider (clocks) | SDA Hold (clocks) | SCL Hold (start) | SCL Hold (stop) |
|---|---|---|---|---|
| 06 | 34 | 10 | 13 | 18 |
| 07 | 40 | 10 | 16 | 21 |
| 08 | 28 | 7 | 10 | 15 |
| 09 | 32 | 7 | 12 | 17 |
| 0A | 36 | 9 | 14 | 19 |
| 0B | 40 | 9 | 16 | 21 |
| 0C | 44 | 11 | 18 | 23 |
| 0D | 48 | 11 | 20 | 25 |
| 0E | 56 | 13 | 24 | 29 |
| 0F | 68 | 13 | 30 | 35 |
| 10 | 48 | 9 | 18 | 25 |
| 11 | 56 | 9 | 22 | 29 |
| 12 | 64 | 13 | 26 | 33 |
| 13 | 72 | 13 | 30 | 37 |
| 14 | 80 | 17 | 34 | 41 |
| 15 | 88 | 17 | 38 | 45 |
| 16 | 104 | 21 | 46 | 53 |
| 17 | 128 | 21 | 58 | 65 |
| 18 | 80 | 9 | 38 | 41 |
| 19 | 96 | 9 | 46 | 49 |
| 1A | 112 | 17 | 54 | 57 |
| 1B | 128 | 17 | 62 | 65 |
| 1C | 144 | 25 | 70 | 73 |
| 1D | 160 | 25 | 78 | 81 |
| 1E | 192 | 33 | 94 | 97 |
| 1F | 240 | 33 | 118 | 121 |
| 20 | 160 | 17 | 78 | 81 |
| 21 | 192 | 17 | 94 | 97 |
| 22 | 224 | 33 | 110 | 113 |
| 23 | 256 | 33 | 126 | 129 |
| 24 | 288 | 49 | 142 | 145 |
| 25 | 320 | 49 | 158 | 161 |
| 26 | 384 | 65 | 190 | 193 |
| 27 | 480 | 65 | 238 | 241 |
| 28 | 320 | 33 | 158 | 161 |
| 29 | 384 | 33 | 190 | 193 |
| 2A | 448 | 65 | 222 | 225 |
| 2B | 512 | 65 | 254 | 257 |
| 2C | 576 | 97 | 286 | 289 |
| 2D | 640 | 97 | 318 | 321 |
| 2E | 768 | 129 | 382 | 385 |
| 2F | 960 | 129 | 478 | 481 |

**Table 15-7. IIC Divider and Hold Values (Sheet 3 of 6)**

| IBC[7:0] (hex) | SCL Divider (clocks) | SDA Hold (clocks) | SCL Hold (start) | SCL Hold (stop) |
|---|---|---|---|---|
| 30 | 640 | 65 | 318 | 321 |
| 31 | 768 | 65 | 382 | 385 |
| 32 | 896 | 129 | 446 | 449 |
| 33 | 1024 | 129 | 510 | 513 |
| 34 | 1152 | 193 | 574 | 577 |
| 35 | 1280 | 193 | 638 | 641 |
| 36 | 1536 | 257 | 766 | 769 |
| 37 | 1920 | 257 | 958 | 961 |
| 38 | 1280 | 129 | 638 | 641 |
| 39 | 1536 | 129 | 766 | 769 |
| 3A | 1792 | 257 | 894 | 897 |
| 3B | 2048 | 257 | 1022 | 1025 |
| 3C | 2304 | 385 | 1150 | 1153 |
| 3D | 2560 | 385 | 1278 | 1281 |
| 3E | 3072 | 513 | 1534 | 1537 |
| 3F | 3840 | 513 | 1918 | 1921 |
| **MUL=2** | | | | |
| 40 | 40 | 14 | 12 | 22 |
| 41 | 44 | 14 | 14 | 24 |
| 42 | 48 | 16 | 16 | 26 |
| 43 | 52 | 16 | 18 | 28 |
| 44 | 56 | 18 | 20 | 30 |
| 45 | 60 | 18 | 22 | 32 |
| 46 | 68 | 20 | 26 | 36 |
| 47 | 80 | 20 | 32 | 42 |
| 48 | 56 | 14 | 20 | 30 |
| 49 | 64 | 14 | 24 | 34 |
| 4A | 72 | 18 | 28 | 38 |
| 4B | 80 | 18 | 32 | 42 |
| 4C | 88 | 22 | 36 | 46 |
| 4D | 96 | 22 | 40 | 50 |
| 4E | 112 | 26 | 48 | 58 |
| 4F | 136 | 26 | 60 | 70 |
| 50 | 96 | 18 | 36 | 50 |
| 51 | 112 | 18 | 44 | 58 |
| 52 | 128 | 26 | 52 | 66 |
| 53 | 144 | 26 | 60 | 74 |
| 54 | 160 | 34 | 68 | 82 |
| 55 | 176 | 34 | 76 | 90 |
| 56 | 208 | 42 | 92 | 106 |
| 57 | 256 | 42 | 116 | 130 |
| 58 | 160 | 18 | 76 | 82 |

**Table 15-7. IIC Divider and Hold Values (Sheet 4 of 6)**

| IBC[7:0] (hex) | SCL Divider (clocks) | SDA Hold (clocks) | SCL Hold (start) | SCL Hold (stop) |
|---|---|---|---|---|
| 59 | 192 | 18 | 92 | 98 |
| 5A | 224 | 34 | 108 | 114 |
| 5B | 256 | 34 | 124 | 130 |
| 5C | 288 | 50 | 140 | 146 |
| 5D | 320 | 50 | 156 | 162 |
| 5E | 384 | 66 | 188 | 194 |
| 5F | 480 | 66 | 236 | 242 |
| 60 | 320 | 34 | 156 | 162 |
| 61 | 384 | 34 | 188 | 194 |
| 62 | 448 | 66 | 220 | 226 |
| 63 | 512 | 66 | 252 | 258 |
| 64 | 576 | 98 | 284 | 290 |
| 65 | 640 | 98 | 316 | 322 |
| 66 | 768 | 130 | 380 | 386 |
| 67 | 960 | 130 | 476 | 482 |
| 68 | 640 | 66 | 316 | 322 |
| 69 | 768 | 66 | 380 | 386 |
| 6A | 896 | 130 | 444 | 450 |
| 6B | 1024 | 130 | 508 | 514 |
| 6C | 1152 | 194 | 572 | 578 |
| 6D | 1280 | 194 | 636 | 642 |
| 6E | 1536 | 258 | 764 | 770 |
| 6F | 1920 | 258 | 956 | 962 |
| 70 | 1280 | 130 | 636 | 642 |
| 71 | 1536 | 130 | 764 | 770 |
| 72 | 1792 | 258 | 892 | 898 |
| 73 | 2048 | 258 | 1020 | 1026 |
| 74 | 2304 | 386 | 1148 | 1154 |
| 75 | 2560 | 386 | 1276 | 1282 |
| 76 | 3072 | 514 | 1532 | 1538 |
| 77 | 3840 | 514 | 1916 | 1922 |
| 78 | 2560 | 258 | 1276 | 1282 |
| 79 | 3072 | 258 | 1532 | 1538 |
| 7A | 3584 | 514 | 1788 | 1794 |
| 7B | 4096 | 514 | 2044 | 2050 |
| 7C | 4608 | 770 | 2300 | 2306 |
| 7D | 5120 | 770 | 2556 | 2562 |
| 7E | 6144 | 1026 | 3068 | 3074 |
| 7F | 7680 | 1026 | 3836 | 3842 |
| **MUL=4** | | | | |
| 80 | 72 | 28 | 24 | 44 |
| 81 | 80 | 28 | 28 | 48 |

**Table 15-7. IIC Divider and Hold Values (Sheet 5 of 6)**

| IBC[7:0] (hex) | SCL Divider (clocks) | SDA Hold (clocks) | SCL Hold (start) | SCL Hold (stop) |
|---|---|---|---|---|
| 82 | 88 | 32 | 32 | 52 |
| 83 | 96 | 32 | 36 | 56 |
| 84 | 104 | 36 | 40 | 60 |
| 85 | 112 | 36 | 44 | 64 |
| 86 | 128 | 40 | 52 | 72 |
| 87 | 152 | 40 | 64 | 84 |
| 88 | 112 | 28 | 40 | 60 |
| 89 | 128 | 28 | 48 | 68 |
| 8A | 144 | 36 | 56 | 76 |
| 8B | 160 | 36 | 64 | 84 |
| 8C | 176 | 44 | 72 | 92 |
| 8D | 192 | 44 | 80 | 100 |
| 8E | 224 | 52 | 96 | 116 |
| 8F | 272 | 52 | 120 | 140 |
| 90 | 192 | 36 | 72 | 100 |
| 91 | 224 | 36 | 88 | 116 |
| 92 | 256 | 52 | 104 | 132 |
| 93 | 288 | 52 | 120 | 148 |
| 94 | 320 | 68 | 136 | 164 |
| 95 | 352 | 68 | 152 | 180 |
| 96 | 416 | 84 | 184 | 212 |
| 97 | 512 | 84 | 232 | 260 |
| 98 | 320 | 36 | 152 | 164 |
| 99 | 384 | 36 | 184 | 196 |
| 9A | 448 | 68 | 216 | 228 |
| 9B | 512 | 68 | 248 | 260 |
| 9C | 576 | 100 | 280 | 292 |
| 9D | 640 | 100 | 312 | 324 |
| 9E | 768 | 132 | 376 | 388 |
| 9F | 960 | 132 | 472 | 484 |
| A0 | 640 | 68 | 312 | 324 |
| A1 | 768 | 68 | 376 | 388 |
| A2 | 896 | 132 | 440 | 452 |
| A3 | 1024 | 132 | 504 | 516 |
| A4 | 1152 | 196 | 568 | 580 |
| A5 | 1280 | 196 | 632 | 644 |
| A6 | 1536 | 260 | 760 | 772 |
| A7 | 1920 | 260 | 952 | 964 |
| A8 | 1280 | 132 | 632 | 644 |
| A9 | 1536 | 132 | 760 | 772 |
| AA | 1792 | 260 | 888 | 900 |
| AB | 2048 | 260 | 1016 | 1028 |

**Table 15-7. IIC Divider and Hold Values (Sheet 6 of 6)**

| IBC[7:0] (hex) | SCL Divider (clocks) | SDA Hold (clocks) | SCL Hold (start) | SCL Hold (stop) |
|---|---|---|---|---|
| AC | 2304 | 388 | 1144 | 1156 |
| AD | 2560 | 388 | 1272 | 1284 |
| AE | 3072 | 516 | 1528 | 1540 |
| AF | 3840 | 516 | 1912 | 1924 |
| B0 | 2560 | 260 | 1272 | 1284 |
| B1 | 3072 | 260 | 1528 | 1540 |
| B2 | 3584 | 516 | 1784 | 1796 |
| B3 | 4096 | 516 | 2040 | 2052 |
| B4 | 4608 | 772 | 2296 | 2308 |
| B5 | 5120 | 772 | 2552 | 2564 |
| B6 | 6144 | 1028 | 3064 | 3076 |
| B7 | 7680 | 1028 | 3832 | 3844 |
| B8 | 5120 | 516 | 2552 | 2564 |
| B9 | 6144 | 516 | 3064 | 3076 |
| BA | 7168 | 1028 | 3576 | 3588 |
| BB | 8192 | 1028 | 4088 | 4100 |
| BC | 9216 | 1540 | 4600 | 4612 |
| BD | 10240 | 1540 | 5112 | 5124 |
| BE | 12288 | 2052 | 6136 | 6148 |
| BF | 15360 | 2052 | 7672 | 7684 |

### 15.3.1.3 IIC Control Register (IBCR)

Module Base + 0x0002

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | IBEN | IBIE | MS/SL | Tx/Rx | TXAK | 0 | 0 | IBSWAI |
| W | | | | | | RSTA | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented or Reserved

**Figure 15-6. IIC Bus Control Register (IBCR)**

Read and write anytime

**Table 15-8. IBCR Field Descriptions**

| Field | Description |
|---|---|
| 7<br>IBEN | **I-Bus Enable** — This bit controls the software reset of the entire IIC bus module.<br>0 The module is reset and disabled. This is the power-on reset situation. When low the interface is held in reset but registers can be accessed<br>1 The IIC bus module is enabled.This bit must be set before any other IBCR bits have any effect<br>If the IIC bus module is enabled in the middle of a byte transfer the interface behaves as follows: slave mode ignores the current transfer on the bus and starts operating whenever a subsequent start condition is detected. Master mode will not be aware that the bus is busy, hence if a start cycle is initiated then the current bus cycle may become corrupt. This would ultimately result in either the current bus master or the IIC bus module losing arbitration, after which bus operation would return to normal. |
| 6<br>IBIE | **I-Bus Interrupt Enable**<br>0 Interrupts from the IIC bus module are disabled. Note that this does not clear any currently pending interrupt condition<br>1 Interrupts from the IIC bus module are enabled. An IIC bus interrupt occurs provided the IBIF bit in the status register is also set. |
| 5<br>MS/SL | **Master/Slave Mode Select Bit** — Upon reset, this bit is cleared. When this bit is changed from 0 to 1, a START signal is generated on the bus, and the master mode is selected. When this bit is changed from 1 to 0, a STOP signal is generated and the operation mode changes from master to slave.A STOP signal should only be generated if the IBIF flag is set. MS/$\overline{SL}$ is cleared without generating a STOP signal when the master loses arbitration.<br>0 Slave Mode<br>1 Master Mode |
| 4<br>Tx/Rx | **Transmit/Receive Mode Select Bit** — This bit selects the direction of master and slave transfers. When addressed as a slave this bit should be set by software according to the SRW bit in the status register. In master mode this bit should be set according to the type of transfer required. Therefore, for address cycles, this bit will always be high.<br>0 Receive<br>1 Transmit |
| 3<br>TXAK | **Transmit Acknowledge Enable** — This bit specifies the value driven onto SDA during data acknowledge cycles for both master and slave receivers. The IIC module will always acknowledge address matches, provided it is enabled, regardless of the value of TXAK. Note that values written to this bit are only used when the IIC bus is a receiver, not a transmitter.<br>0 An acknowledge signal will be sent out to the bus at the 9th clock bit after receiving one byte data<br>1 No acknowledge signal response is sent (i.e., acknowledge bit = 1) |
| 2<br>RSTA | **Repeat Start** — Writing a 1 to this bit will generate a repeated START condition on the bus, provided it is the current bus master. This bit will always be read as a low. Attempting a repeated start at the wrong time, if the bus is owned by another master, will result in loss of arbitration.<br>1 Generate repeat start cycle |
| 1<br>RESERVED | **Reserved** — Bit 1 of the IBCR is reserved for future compatibility. This bit will always read 0. |
| 0<br>IBSWAI | **I Bus Interface Stop in Wait Mode**<br>0 IIC bus module clock operates normally<br>1 Halt IIC bus module clock generation in wait mode |

Wait mode is entered via execution of a CPU WAI instruction. In the event that the IBSWAI bit is set, all clocks internal to the IIC will be stopped and any transmission currently in progress will halt.If the CPU were woken up by a source other than the IIC module, then clocks would restart and the IIC would resume from where was during the previous transmission. It is not possible for the IIC to wake up the CPU when its internal clocks are stopped.

If it were the case that the IBSWAI bit was cleared when the WAI instruction was executed, the IIC internal clocks and interface would remain alive, continuing the operation which was currently underway. It is also possible to configure the IIC such that it will wake up the CPU via an interrupt at the conclusion of the current operation. See the discussion on the IBIF and IBIE bits in the IBSR and IBCR, respectively.

### 15.3.1.4 IIC Status Register (IBSR)

Module Base + 0x0003

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | TCF | IAAS | IBB | IBAL | 0 | SRW | IBIF | RXAK |
| W | | | | | | | | |
| Reset | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented or Reserved

**Figure 15-7. IIC Bus Status Register (IBSR)**

This status register is read-only with exception of bit 1 (IBIF) and bit 4 (IBAL), which are software clearable.

**Table 15-9. IBSR Field Descriptions**

| Field | Description |
|---|---|
| 7<br>TCF | **Data Transferring Bit** — While one byte of data is being transferred, this bit is cleared. It is set by the falling edge of the 9th clock of a byte transfer. Note that this bit is only valid during or immediately following a transfer to the IIC module or from the IIC module.<br>0 Transfer in progress<br>1 Transfer complete |
| 6<br>IAAS | **Addressed as a Slave Bit** — When its own specific address (I-bus address register) is matched with the calling address or it receives the general call address with GCEN== 1,this bit is set.The CPU is interrupted provided the IBIE is set.Then the CPU needs to check the SRW bit and set its Tx/$\overline{\text{Rx}}$ mode accordingly.Writing to the I-bus control register clears this bit.<br>0 Not addressed<br>1 Addressed as a slave |
| 5<br>IBB | Bus Busy Bit<br>0 This bit indicates the status of the bus. When a START signal is detected, the IBB is set. If a STOP signal is detected, IBB is cleared and the bus enters idle state.<br>1 Bus is busy |
| 4<br>IBAL | **Arbitration Lost** — The arbitration lost bit (IBAL) is set by hardware when the arbitration procedure is lost. Arbitration is lost in the following circumstances:<br>1. SDA sampled low when the master drives a high during an address or data transmit cycle.<br>2. SDA sampled low when the master drives a high during the acknowledge bit of a data receive cycle.<br>3. A start cycle is attempted when the bus is busy.<br>4. A repeated start cycle is requested in slave mode.<br>5. A stop condition is detected when the master did not request it.<br>This bit must be cleared by software, by writing a one to it. A write of 0 has no effect on this bit. |
| 3<br>RESERVED | **Reserved** — Bit 3 of IBSR is reserved for future use. A read operation on this bit will return 0. |

**Table 15-9. IBSR Field Descriptions (continued)**

| Field | Description |
|---|---|
| 2<br>SRW | **Slave Read/Write** — When IAAS is set this bit indicates the value of the R/W command bit of the calling address sent from the master<br>This bit is only valid when the I-bus is in slave mode, a complete address transfer has occurred with an address match and no other transfers have been initiated.<br>Checking this bit, the CPU can select slave transmit/receive mode according to the command of the master.<br>0 Slave receive, master writing to slave<br>1 Slave transmit, master reading from slave |
| 1<br>IBIF | **I-Bus Interrupt** — The IBIF bit is set when one of the following conditions occurs:<br>— Arbitration lost (IBAL bit set)<br>— Data transfer complete (TCF bit set)<br>— Addressed as slave (IAAS bit set)<br>It will cause a processor interrupt request if the IBIE bit is set. This bit must be cleared by software, writing a one to it. A write of 0 has no effect on this bit. |
| 0<br>RXAK | **Received Acknowledge** — The value of SDA during the acknowledge bit of a bus cycle. If the received acknowledge bit (RXAK) is low, it indicates an acknowledge signal has been received after the completion of 8 bits data transmission on the bus. If RXAK is high, it means no acknowledge signal is detected at the 9th clock.<br>0 Acknowledge received<br>1 No acknowledge received |

### 15.3.1.5 IIC Data I/O Register (IBDR)

Module Base + 0x0004

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 15-8. IIC Bus Data I/O Register (IBDR)**

In master transmit mode, when data is written to the IBDR a data transfer is initiated. The most significant bit is sent first. In master receive mode, reading this register initiates next byte data receiving. In slave mode, the same functions are available after an address match has occurred.Note that the Tx/Rx bit in the IBCR must correctly reflect the desired direction of transfer in master and slave modes for the transmission to begin. For instance, if the IIC is configured for master transmit but a master receive is desired, then reading the IBDR will not initiate the receive.

Reading the IBDR will return the last byte received while the IIC is configured in either master receive or slave receive modes. The IBDR does not reflect every byte that is transmitted on the IIC bus, nor can software verify that a byte has been written to the IBDR correctly by reading it back.

In master transmit mode, the first byte of data written to IBDR following assertion of MS/$\overline{\text{SL}}$ is used for the address transfer and should com.prise of the calling address (in position D7:D1) concatenated with the required R/$\overline{\text{W}}$ bit (in position D0).

### 15.3.1.6 IIC Control Register 2(IBCR2)

Module Base + 0x0005

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | GCEN | ADTYPE | 0 | 0 | 0 | ADR10 | ADR9 | ADR8 |
| W | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 15-9. IIC Bus Control Register 2(IBCR2)**

This register contains the variables used in general call and in ten-bit address.

Read and write anytime

**Table 15-10. IBCR2 Field Descriptions**

| Field | Description |
|---|---|
| 7<br>GCEN | **General Call Enable**.<br>0 General call is disabled. The module dont receive any general call data and address.<br>1 enable general call. It indicates that the module can receive address and any data. |
| 6<br>ADTYPE | **Address Type—** This bit selects the address length. The variable must be configured correctly before IIC enters slave mode.<br>0 7-bit address<br>1 10-bit address |
| 5,4,3<br>RESERVED | **Reserved** — Bit 5,4 and 3 of the IBCR2 are reserved for future compatibility. These bits will always read 0. |
| 2:0<br>ADR[10:8] | **Slave Address [10:8]** —These 3 bits represent the MSB of the 10-bit address when address type is asserted (ADTYPE = 1). |

## 15.4 Functional Description

This section provides a complete functional description of the IICV3.

### 15.4.1 I-Bus Protocol

The IIC bus system uses a serial data line (SDA) and a serial clock line (SCL) for data transfer. All devices connected to it must have open drain or open collector outputs. Logic AND function is exercised on both lines with external pull-up resistors. The value of these resistors is system dependent.

Normally, a standard communication is composed of four parts: START signal, slave address transmission, data transfer and STOP signal. They are described briefly in the following sections and illustrated in Figure 15-10.

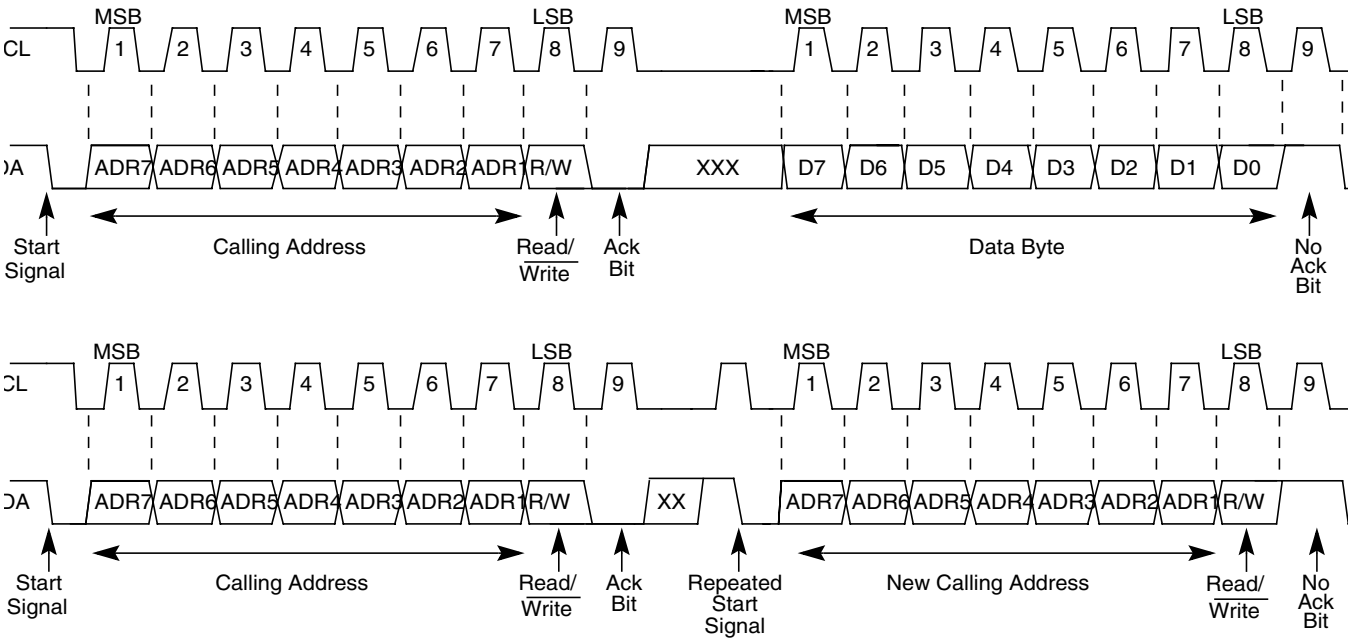



**Figure 15-10. IIC-Bus Transmission Signals**

### 15.4.1.1 START Signal

When the bus is free, i.e. no master device is engaging the bus (both SCL and SDA lines are at logical high), a master may initiate communication by sending a START signal.As shown in Figure 15-10, a START signal is defined as a high-to-low transition of SDA while SCL is high. This signal denotes the beginning of a new data transfer (each data transfer may contain several bytes of data) and brings all slaves out of their idle states.

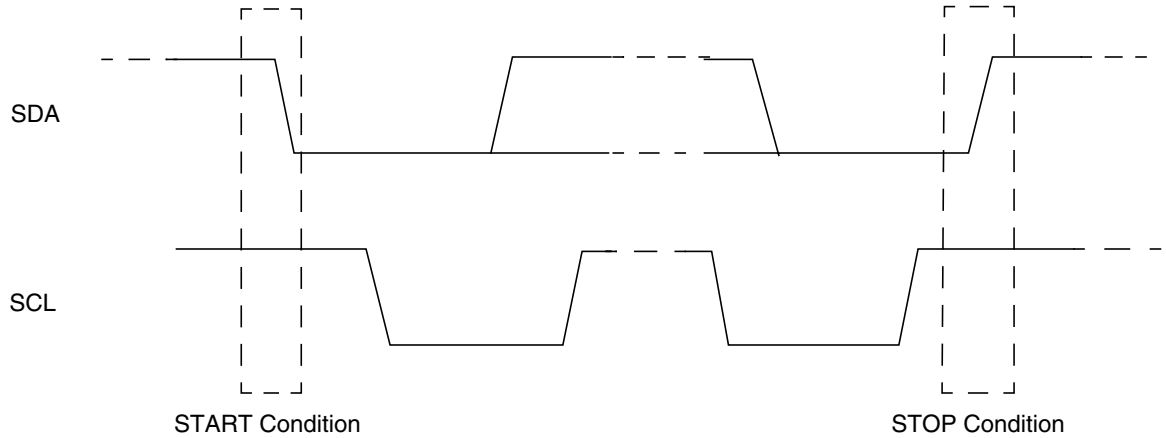



**Figure 15-11. Start and Stop Conditions**

#### 15.4.1.2 Slave Address Transmission

The first byte of data transfer immediately after the START signal is the slave address transmitted by the master. This is a seven-bit calling address followed by a R/W bit. The R/W bit tells the slave the desired direction of data transfer.

1 = Read transfer, the slave transmits data to the master.
0 = Write transfer, the master transmits data to the slave.

If the calling address is 10-bit, another byte is followed by the first byte.Only the slave with a calling address that matches the one transmitted by the master will respond by sending back an acknowledge bit. This is done by pulling the SDA low at the 9th clock (see Figure 15-10).

No two slaves in the system may have the same address. If the IIC bus is master, it must not transmit an address that is equal to its own slave address. The IIC bus cannot be master and slave at the same time.However, if arbitration is lost during an address cycle the IIC bus will revert to slave mode and operate correctly even if it is being addressed by another master.

#### 15.4.1.3 Data Transfer

As soon as successful slave addressing is achieved, the data transfer can proceed byte-by-byte in a direction specified by the R/W bit sent by the calling master

All transfers that come after an address cycle are referred to as data transfers, even if they carry sub-address information for the slave device.

Each data byte is 8 bits long. Data may be changed only while SCL is low and must be held stable while SCL is high as shown in Figure 15-10. There is one clock pulse on SCL for each data bit, the MSB being transferred first. Each data byte has to be followed by an acknowledge bit, which is signalled from the receiving device by pulling the SDA low at the ninth clock. So one complete data byte transfer needs nine clock pulses.

If the slave receiver does not acknowledge the master, the SDA line must be left high by the slave. The master can then generate a stop signal to abort the data transfer or a start signal (repeated start) to commence a new calling.

If the master receiver does not acknowledge the slave transmitter after a byte transmission, it means 'end of data' to the slave, so the slave releases the SDA line for the master to generate STOP or START signal.Note in order to release the bus correctly,after no-acknowledge to the master,the slave must be immediately switched to receiver and a following dummy reading of the IBDR is necessary.

#### 15.4.1.4 STOP Signal

The master can terminate the communication by generating a STOP signal to free the bus. However, the master may generate a START signal followed by a calling command without generating a STOP signal first. This is called repeated START. A STOP signal is defined as a low-to-high transition of SDA while SCL at logical 1 (see Figure 15-10).

The master can generate a STOP even if the slave has generated an acknowledge at which point the slave must release the bus.

### 15.4.1.5 Repeated START Signal

As shown in Figure 15-10, a repeated START signal is a START signal generated without first generating a STOP signal to terminate the communication. This is used by the master to communicate with another slave or with the same slave in different mode (transmit/receive mode) without releasing the bus.

### 15.4.1.6 Arbitration Procedure

The Inter-IC bus is a true multi-master bus that allows more than one master to be connected on it. If two or more masters try to control the bus at the same time, a clock synchronization procedure determines the bus clock, for which the low period is equal to the longest clock low period and the high is equal to the shortest one among the masters. The relative priority of the contending masters is determined by a data arbitration procedure, a bus master loses arbitration if it transmits logic 1 while another master transmits logic 0. The losing masters immediately switch over to slave receive mode and stop driving SDA output. In this case the transition from master to slave mode does not generate a STOP condition. Meanwhile, a status bit is set by hardware to indicate loss of arbitration.

### 15.4.1.7 Clock Synchronization

Because wire-AND logic is performed on SCL line, a high-to-low transition on SCL line affects all the devices connected on the bus. The devices start counting their low period and as soon as a device's clock has gone low, it holds the SCL line low until the clock high state is reached.However, the change of low to high in this device clock may not change the state of the SCL line if another device clock is within its low period. Therefore, synchronized clock SCL is held low by the device with the longest low period. Devices with shorter low periods enter a high wait state during this time (see Figure 15-11). When all devices concerned have counted off their low period, the synchronized clock SCL line is released and pulled high. There is then no difference between the device clocks and the state of the SCL line and all the devices start counting their high periods.The first device to complete its high period pulls the SCL line low again.

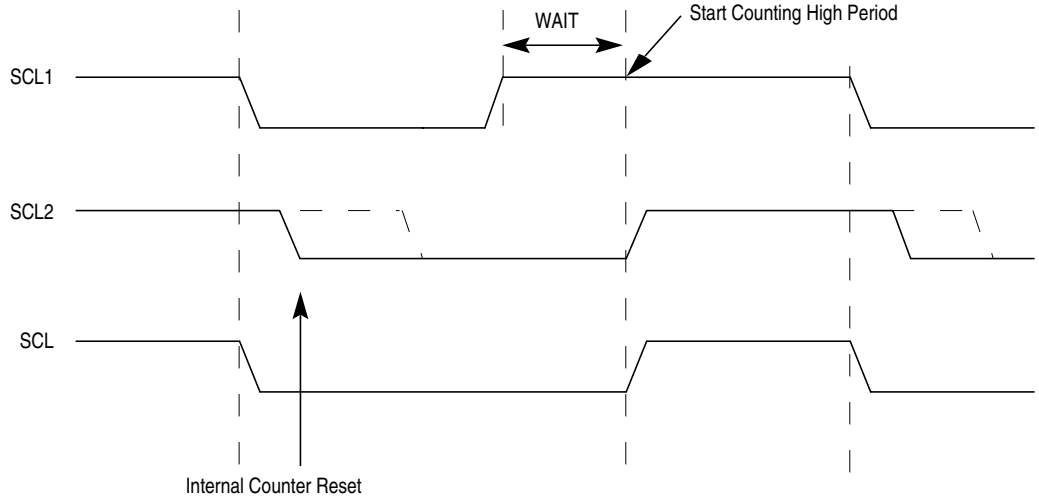



**Figure 15-12. IIC-Bus Clock Synchronization**

### 15.4.1.8 Handshaking

The clock synchronization mechanism can be used as a handshake in data transfer. Slave devices may hold the SCL low after completion of one byte transfer (9 bits). In such case, it halts the bus clock and forces the master clock into wait states until the slave releases the SCL line.

### 15.4.1.9 Clock Stretching

The clock synchronization mechanism can be used by slaves to slow down the bit rate of a transfer. After the master has driven SCL low the slave can drive SCL low for the required period and then release it.If the slave SCL low period is greater than the master SCL low period then the resulting SCL bus signal low period is stretched.

### 15.4.1.10 Ten-bit Address

A ten-bit address is indicated if the first 5 bits of the first address byte are 0x11110. The following rules apply to the first address byte.

**Table 15-11. Definition of Bits in the First Byte**

| SLAVE ADDRESS | R/W BIT | DESCRIPTION |
|---|---|---|
| 0000000 | 0 | General call address |
| 0000010 | x | Reserved for different bus format |
| 0000011 | x | Reserved for future purposes |
| 11111XX | x | Reserved for future purposes |
| 11110XX | x | 10-bit slave addressing |

The address type is identified by ADTYPE. When ADTYPE is 0, 7-bit address is applied. Reversely, the address is 10-bit address.Generally, there are two cases of 10-bit address.See the Fig.1-14 and 1-15.

| S | Slave Add1st 7bits 11110+ADR10+ADR9 | R/W 0 | A1 | Slave Add 2nd byte ADR[8:1] | A2 | Data | A3 |
|---|---|---|---|---|---|---|---|

**Figure 15-13. A master-transmitter addresses a slave-receiver with a 10-bit address**

| S | Slave Add1st 7bits 11110+ADR10+ADR9 | R/W 0 | A1 | Slave Add 2nd byte ADR[8:1] | A2 | Sr | Slave Add 1st 7bits 11110+ADR10+ADR9 | R/W 1 | A3 | Data | A4 |
|---|---|---|---|---|---|---|---|---|---|---|---|

**Figure 15-14. A master-receiver addresses a slave-transmitter with a 10-bit address**

In the figure 1-15,the first two bytes are the similar to figure1-14.After the repeated START(Sr),the first slave address is transmitted again, but the R/W is 1, meaning that the slave is acted as a transmitter.

### 15.4.1.11 General Call Address

To broadcast using a general call, a device must first generate the general call address($00), then after receiving acknowledge, it must transmit data.

In communication, as a slave device, provided the GCEN is asserted, a device acknowledges the broadcast and receives data until the GCEN is disabled or the master device releases the bus or generates a new

transfer. In the broadcast, slaves always act as receivers. In general call, IAAS is also used to indicate the address match.

In order to distinguish whether the address match is the normal address match or the general call address match, IBDR should be read after the address byte has been received. If the data is $00, the match is general call address match. The meaning of the general call address is always specified in the first data byte and must be dealt with by S/W, the IIC hardware does not decode and process the first data byte.

When one byte transfer is done, the received data can be read from IBDR. The user can control the procedure by enabling or disabling GCEN.

### 15.4.2 Operation in Run Mode

This is the basic mode of operation.

### 15.4.3 Operation in Wait Mode

IIC operation in wait mode can be configured. Depending on the state of internal bits, the IIC can operate normally when the CPU is in wait mode or the IIC clock generation can be turned off and the IIC module enters a power conservation state during wait mode. In the later case, any transmission or reception in progress stops at wait mode entry.

### 15.4.4 Operation in Stop Mode

The IIC is inactive in stop mode for reduced power consumption. The STOP instruction does not affect IIC register states.

## 15.5 Resets

The reset state of each individual bit is listed in Section 15.3, “Memory Map and Register Definition,” which details the registers and their bit-fields.

## 15.6 Interrupts

IICV3 uses only one interrupt vector.

**Table 15-12. Interrupt Summary**

| Interrupt | Offset | Vector | Priority | Source | Description |
|---|---|---|---|---|---|
| IIC Interrupt | — | — | — | IBAL, TCF, IAAS bits in IBSR register | When either of IBAL, TCF or IAAS bits is set may cause an interrupt based on arbitration lost, transfer complete or address detect conditions |

Internally there are three types of interrupts in IIC. The interrupt service routine can determine the interrupt type by reading the status register.

IIC Interrupt can be generated on

1. Arbitration lost condition (IBAL bit set)

2. Byte transfer condition (TCF bit set)
3. Address detect condition (IAAS bit set)

The IIC interrupt is enabled by the IBIE bit in the IIC control register. It must be cleared by writing 0 to the IBF bit in the interrupt service routine.

# 15.7 Application Information

## 15.7.1 IIC Programming Examples

### 15.7.1.1 Initialization Sequence

Reset will put the IIC bus control register to its default status. Before the interface can be used to transfer serial data, an initialization procedure must be carried out, as follows:

1. Update the frequency divider register (IBFD) and select the required division ratio to obtain SCL frequency from system clock.
2. Update the ADTYPE of IBCR2 to define the address length, 7 bits or 10 bits.
3. Update the IIC bus address register (IBAD) to define its slave address. If 10-bit address is applied IBCR2 should be updated to define the rest bits of address.
4. Set the IBEN bit of the IIC bus control register (IBCR) to enable the IIC interface system.
5. Modify the bits of the IIC bus control register (IBCR) to select master/slave mode, transmit/receive mode and interrupt enable or not.
6. If supported general call, the GCEN in IBCR2 should be asserted.

### 15.7.1.2 Generation of START

After completion of the initialization procedure, serial data can be transmitted by selecting the 'master transmitter' mode. If the device is connected to a multi-master bus system, the state of the IIC bus busy bit (IBB) must be tested to check whether the serial bus is free.

If the bus is free (IBB=0), the start condition and the first byte (the slave address) can be sent. The data written to the data register comprises the slave calling address and the LSB set to indicate the direction of transfer required from the slave.

The bus free time (i.e., the time between a STOP condition and the following START condition) is built into the hardware that generates the START cycle. Depending on the relative frequencies of the system clock and the SCL period it may be necessary to wait until the IIC is busy after writing the calling address to the IBDR before proceeding with the following instructions. This is illustrated in the following example.

An example of a program which generates the START signal and transmits the first byte of data (slave address) is shown below:

```
CHFLAG      BRSET   IBSR,#$20,*     ;WAIT FOR IBB FLAG TO CLEAR
TXSTART     BSET    IBCR,#$30       ;SET TRANSMIT AND MASTER MODE;i.e. GENERATE START CONDITION
IBFREE      BRCLR   IBSR,#$20,*     ;WAIT FOR IBB FLAG TO SET
```

```
            MOVB    CALLING,IBDR    ;TRANSMIT THE CALLING ADDRESS, D0=R/W
IBFREE      BRCLR   IBSR,#$20,*     ;WAIT FOR IBB FLAG TO SET
```

### 15.7.1.3 Post-Transfer Software Response

Transmission or reception of a byte will set the data transferring bit (TCF) to 1, which indicates one byte communication is finished. The IIC bus interrupt bit (IBIF) is set also; an interrupt will be generated if the interrupt function is enabled during initialization by setting the IBIE bit. Software must clear the IBIF bit in the interrupt routine first. The TCF bit will be cleared by reading from the IIC bus data I/O register (IBDR) in receive mode or writing to IBDR in transmit mode.

Software may service the IIC I/O in the main program by monitoring the IBIF bit if the interrupt function is disabled. Note that polling should monitor the IBIF bit rather than the TCF bit because their operation is different when arbitration is lost.

Note that when an interrupt occurs at the end of the address cycle the master will always be in transmit mode, i.e. the address is transmitted. If master receive mode is required, indicated by R/W bit in IBDR, then the Tx/Rx bit should be toggled at this stage.

During slave mode address cycles (IAAS=1), the SRW bit in the status register is read to determine the direction of the subsequent transfer and the Tx/Rx bit is programmed accordingly.For slave mode data cycles (IAAS=0) the SRW bit is not valid, the Tx/Rx bit in the control register should be read to determine the direction of the current transfer.

The following is an example of a software response by a 'master transmitter' in the interrupt routine.

```
ISR         BCLR    IBSR,#$02           ;CLEAR THE IBIF FLAG
            BRCLR   IBCR,#$20,SLAVE     ;BRANCH IF IN SLAVE MODE
            BRCLR   IBCR,#$10,RECEIVE   ;BRANCH IF IN RECEIVE MODE
            BRSET   IBSR,#$01,END       ;IF NO ACK, END OF TRANSMISSION
TRANSMIT    MOVB    DATABUF,IBDR        ;TRANSMIT NEXT BYTE OF DATA
```

### 15.7.1.4 Generation of STOP

A data transfer ends with a STOP signal generated by the 'master' device. A master transmitter can simply generate a STOP signal after all the data has been transmitted. The following is an example showing how a stop condition is generated by a master transmitter.

```
MASTX       TST     TXCNT           ;GET VALUE FROM THE TRANSMITING COUNTER
            BEQ     END             ;END IF NO MORE DATA
            BRSET   IBSR,#$01,END   ;END IF NO ACK
            MOVB    DATABUF,IBDR    ;TRANSMIT NEXT BYTE OF DATA
            DEC     TXCNT           ;DECREASE THE TXCNT
            BRA     EMASTX          ;EXIT
END         BCLR    IBCR,#$20       ;GENERATE A STOP CONDITION
EMASTX      RTI                     ;RETURN FROM INTERRUPT
```

If a master receiver wants to terminate a data transfer, it must inform the slave transmitter by not acknowledging the last byte of data which can be done by setting the transmit acknowledge bit (TXAK)

before reading the 2nd last byte of data. Before reading the last byte of data, a STOP signal must be generated first. The following is an example showing how a STOP signal is generated by a master receiver.

```
MASR    DEC     RXCNT       ;DECREASE THE RXCNT
        BEQ     ENMASR      ;LAST BYTE TO BE READ
        MOVB    RXCNT,D1    ;CHECK SECOND LAST BYTE
        DEC     D1          ;TO BE READ
        BNE     NXMAR       ;NOT LAST OR SECOND LAST
LAMAR   BSET    IBCR,#$08   ;SECOND LAST, DISABLE ACK
                            ;TRANSMITTING
        BRA     NXMAR
ENMASR  BCLR    IBCR,#$20   ;LAST ONE, GENERATE 'STOP' SIGNAL
NXMAR   MOVB    IBDR,RXBUF  ;READ DATA AND STORE
        RTI
```

### 15.7.1.5 Generation of Repeated START

At the end of data transfer, if the master continues to want to communicate on the bus, it can generate another START signal followed by another slave address without first generating a STOP signal. A program example is as shown.

```
RESTART BSET    IBCR,#$04   ;ANOTHER START (RESTART)
        MOVB    CALLING,IBDR ;TRANSMIT THE CALLING ADDRESS;D0=R/W
```

### 15.7.1.6 Slave Mode

In the slave interrupt service routine, the module addressed as slave bit (IAAS) should be tested to check if a calling of its own address has just been received. If IAAS is set, software should set the transmit/receive mode select bit (Tx/Rx bit of IBCR) according to the R/W command bit (SRW). Writing to the IBCR clears the IAAS automatically. Note that the only time IAAS is read as set is from the interrupt at the end of the address cycle where an address match occurred, interrupts resulting from subsequent data transfers will have IAAS cleared. A data transfer may now be initiated by writing information to IBDR, for slave transmits, or dummy reading from IBDR, in slave receive mode. The slave will drive SCL low in-between byte transfers, SCL is released when the IBDR is accessed in the required mode.

In slave transmitter routine, the received acknowledge bit (RXAK) must be tested before transmitting the next byte of data. Setting RXAK means an 'end of data' signal from the master receiver, after which it must be switched from transmitter mode to receiver mode by software. A dummy read then releases the SCL line so that the master can generate a STOP signal.

### 15.7.1.7 Arbitration Lost

If several masters try to engage the bus simultaneously, only one master wins and the others lose arbitration. The devices which lost arbitration are immediately switched to slave receive mode by the hardware. Their data output to the SDA line is stopped, but SCL continues to be generated until the end of the byte during which arbitration was lost. An interrupt occurs at the falling edge of the ninth clock of this transfer with IBAL=1 and MS/SL=0. If one master attempts to start transmission while the bus is being engaged by another master, the hardware will inhibit the transmission; switch the MS/SL bit from 1 to 0 without generating STOP condition; generate an interrupt to CPU and set the IBAL to indicate that the

attempt to engage the bus is failed. When considering these cases, the slave service routine should test the IBAL first and the software should clear the IBAL bit if it is set.

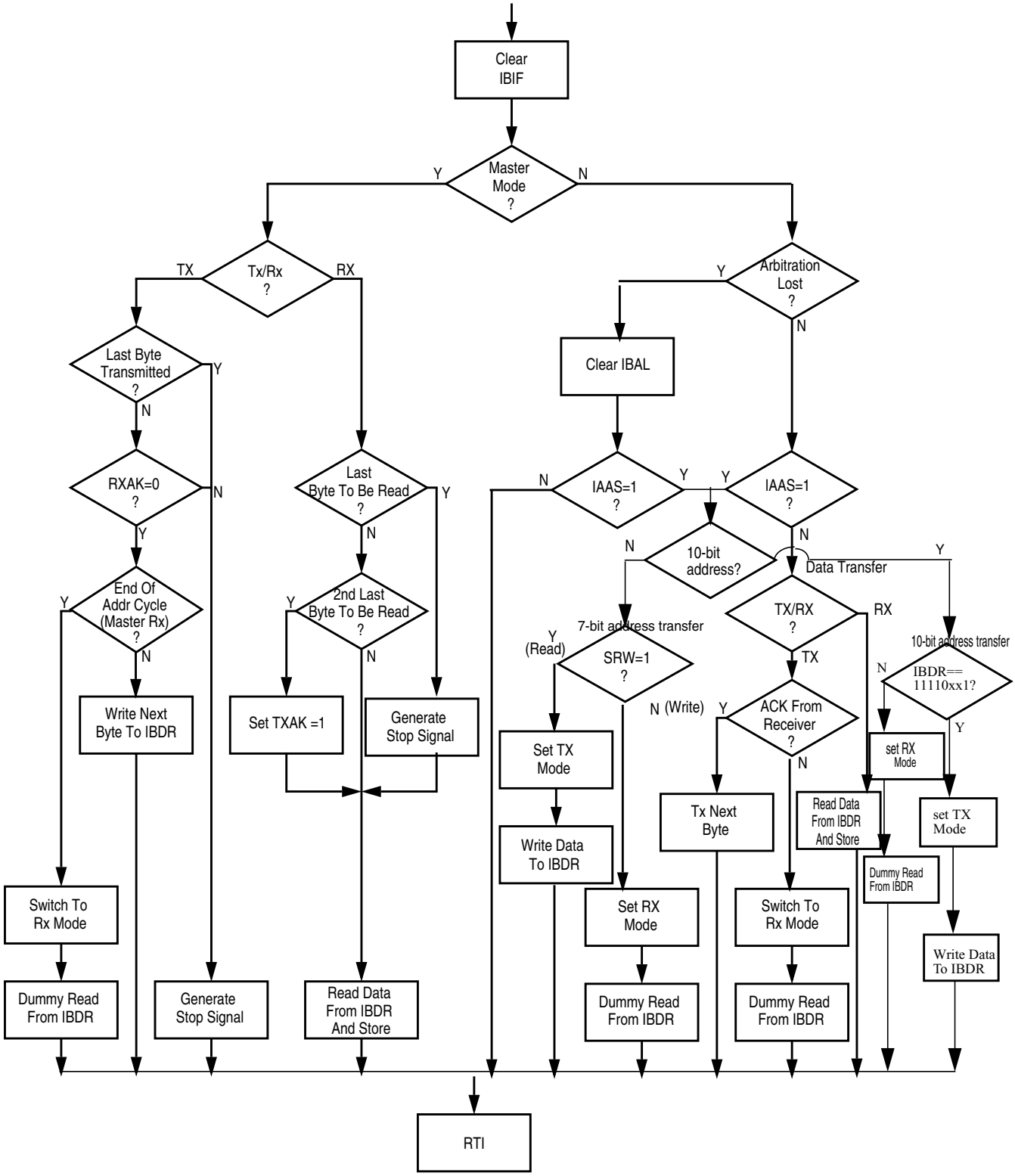



**Figure 15-15. Flow-Chart of Typical IIC Interrupt Routine**

**CAUTION**

When IIC is configured as 10-bit address,the point of the data array in interrupt routine must be reset after it’s addressed.

# Chapter 16
# Freescale's Scalable Controller Area Network (S12MSCANV3)

**Table 16-1. Revision History**

| Revision Number | Revision Date | Sections Affected | Description of Changes |
|---|---|---|---|
| V03.13 | 03 Mar 2011 | Figure 16-4<br>Table 16-3 | • Corrected CANE write restrictions<br>• Removed footnote from RXFRM bit |
| V03.14 | 12 Nov 2012 | Table 16-11 | • Corrected RxWRN and TxWRN threshold values |
| V03.15 | 12 Jan 2013 | Table 16-3<br>Table 16-26<br>Figure 16-37<br>16.1/16-605<br>16.3.2.15/16-626 | • Updated TIME bit description<br>• Added register names to buffer map<br>• Updated TSRH and TSRL read conditions<br>• Updated introduction<br>• Updated CANTXERR and CANRXERR register notes |

## 16.1 Introduction

Freescale's scalable controller area network (S12MSCANV3) definition is based on the MSCAN12 definition, which is the specific implementation of the MSCAN concept targeted for the S12, S12X and S12Z microcontroller families.

The module is a communication controller implementing the CAN 2.0A/B protocol as defined in the Bosch specification dated September 1991. For users to fully understand the MSCAN specification, it is recommended that the Bosch specification be read first to familiarize the reader with the terms and concepts contained within this document.

Though not exclusively intended for automotive applications, CAN protocol is designed to meet the specific requirements of a vehicle serial data bus: real-time processing, reliable operation in the EMI environment of a vehicle, cost-effectiveness, and required bandwidth.

MSCAN uses an advanced buffer arrangement resulting in predictable real-time behavior and simplified application software.

## 16.1.1 Glossary

**Table 16-2. Terminology**

| | |
|---|---|
| ACK | Acknowledge of CAN message |
| CAN | Controller Area Network |
| CRC | Cyclic Redundancy Code |
| EOF | End of Frame |
| FIFO | First-In-First-Out Memory |
| IFS | Inter-Frame Sequence |
| SOF | Start of Frame |
| CPU bus | CPU related read/write data bus |
| CAN bus | CAN protocol related serial bus |
| oscillator clock | Direct clock from external oscillator |
| bus clock | CPU bus related clock |
| CAN clock | CAN protocol related clock |

## 16.1.2 Block Diagram

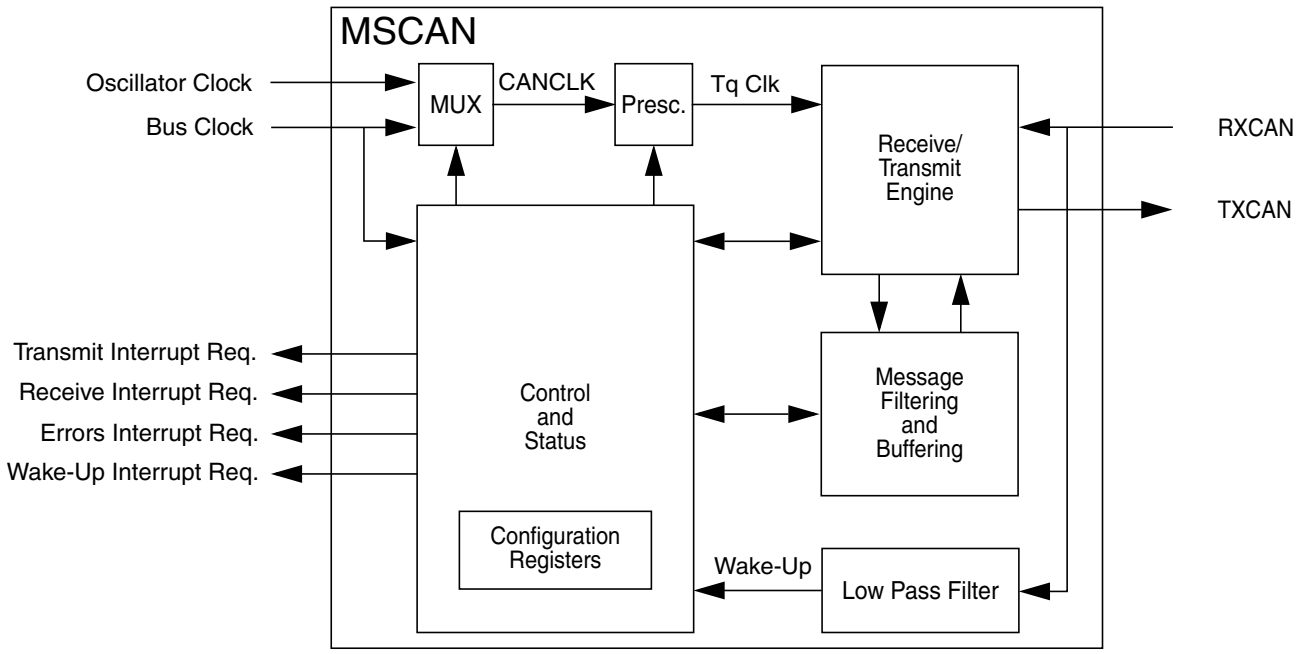



**Figure 16-1. MSCAN Block Diagram**

### 16.1.3 Features

The basic features of the MSCAN are as follows:

- Implementation of the CAN protocol — Version 2.0A/B
  - — Standard and extended data frames
  - — Zero to eight bytes data length
  - — Programmable bit rate up to 1 Mbps[1]
  - — Support for remote frames
- Five receive buffers with FIFO storage scheme
- Three transmit buffers with internal prioritization using a "local priority" concept
- Flexible maskable identifier filter supports two full-size (32-bit) extended identifier filters, or four 16-bit filters, or eight 8-bit filters
- Programmable wake-up functionality with integrated low-pass filter
- Programmable loopback mode supports self-test operation
- Programmable listen-only mode for monitoring of CAN bus
- Programmable bus-off recovery functionality
- Separate signalling and interrupt capabilities for all CAN receiver and transmitter error states (warning, error passive, bus-off)
- Programmable MSCAN clock source either bus clock or oscillator clock
- Internal timer for time-stamping of received and transmitted messages
- Three low-power modes: sleep, power down, and MSCAN enable
- Global initialization of configuration registers

### 16.1.4 Modes of Operation

For a description of the specific MSCAN modes and the module operation related to the system operating modes refer to Section 16.4.4, "Modes of Operation".

1. Depending on the actual bit timing and the clock jitter of the PLL.

## 16.2 External Signal Description

The MSCAN uses two external pins.

**NOTE**

On MCUs with an integrated CAN physical interface (transceiver) the MSCAN interface is connected internally to the transceiver interface. In these cases the external availability of signals TXCAN and RXCAN is optional.

### 16.2.1 RXCAN — CAN Receiver Input Pin

RXCAN is the MSCAN receiver input pin.

### 16.2.2 TXCAN — CAN Transmitter Output Pin

TXCAN is the MSCAN transmitter output pin. The TXCAN output pin represents the logic level on the CAN bus:

0 = Dominant state
1 = Recessive state

### 16.2.3 CAN System

A typical CAN system with MSCAN is shown in Figure 16-2. Each CAN station is connected physically to the CAN bus lines through a transceiver device. The transceiver is capable of driving the large current needed for the CAN bus and has current protection against defective CAN or defective stations.

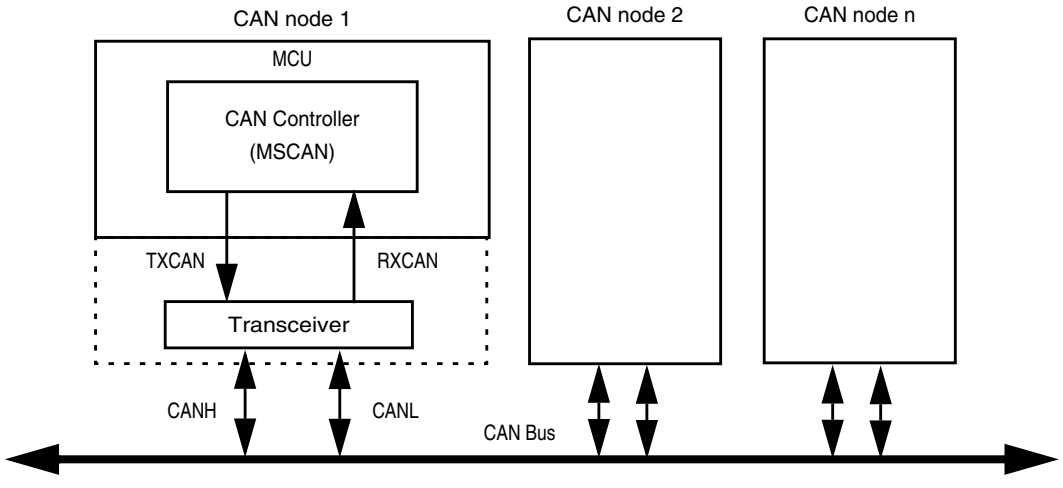



**Figure 16-2. CAN System**

## 16.3 Memory Map and Register Definition

This section provides a detailed description of all registers accessible in the MSCAN.

### 16.3.1 Module Memory Map

Figure 16-3 gives an overview on all registers and their individual bits in the MSCAN memory map. The *register address* results from the addition of *base address* and *address offset*. The *base address* is determined at the MCU level and can be found in the MCU memory map description. The *address offset* is defined at the module level.

The MSCAN occupies 64 bytes in the memory space. The base address of the MSCAN module is determined at the MCU level when the MCU is defined. The register decode map is fixed and begins at the first address of the module address offset.

The detailed register descriptions follow in the order they appear in the register map.

| Register Name | | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|---|
| 0x0000 CANCTL0 | R | RXFRM | RXACT | CSWAI | SYNCH | TIME | WUPE | SLPRQ | INITRQ |
| | W | | | | | | | | |
| 0x0001 CANCTL1 | R | CANE | CLKSRC | LOOPB | LISTEN | BORM | WUPM | SLPAK | INITAK |
| | W | | | | | | | | |
| 0x0002 CANBTR0 | R | SJW1 | SJW0 | BRP5 | BRP4 | BRP3 | BRP2 | BRP1 | BRP0 |
| | W | | | | | | | | |
| 0x0003 CANBTR1 | R | SAMP | TSEG22 | TSEG21 | TSEG20 | TSEG13 | TSEG12 | TSEG11 | TSEG10 |
| | W | | | | | | | | |
| 0x0004 CANRFLG | R | WUPIF | CSCIF | RSTAT1 | RSTAT0 | TSTAT1 | TSTAT0 | OVRIF | RXF |
| | W | | | | | | | | |
| 0x0005 CANRIER | R | WUPIE | CSCIE | RSTATE1 | RSTATE0 | TSTATE1 | TSTATE0 | OVRIE | RXFIE |
| | W | | | | | | | | |
| 0x0006 CANTFLG | R | 0 | 0 | 0 | 0 | 0 | TXE2 | TXE1 | TXE0 |
| | W | | | | | | | | |
| 0x0007 CANTIER | R | 0 | 0 | 0 | 0 | 0 | TXEIE2 | TXEIE1 | TXEIE0 |
| | W | | | | | | | | |
| 0x0008 CANTARQ | R | 0 | 0 | 0 | 0 | 0 | ABTRQ2 | ABTRQ1 | ABTRQ0 |
| | W | | | | | | | | |
| 0x0009 CANTAAK | R | 0 | 0 | 0 | 0 | 0 | ABTAK2 | ABTAK1 | ABTAK0 |
| | W | | | | | | | | |
| 0x000A CANTBSEL | R | 0 | 0 | 0 | 0 | 0 | TX2 | TX1 | TX0 |
| | W | | | | | | | | |
| 0x000B CANIDAC | R | 0 | 0 | IDAM1 | IDAM0 | 0 | IDHIT2 | IDHIT1 | IDHIT0 |
| | W | | | | | | | | |
| 0x000C Reserved | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | W | | | | | | | | |
| 0x000D CANMISC | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | BOHOLD |
| | W | | | | | | | | |
| 0x000E CANRXERR | R | RXERR7 | RXERR6 | RXERR5 | RXERR4 | RXERR3 | RXERR2 | RXERR1 | RXERR0 |
| | W | | | | | | | | |

= Unimplemented or Reserved

**Figure 16-3. MSCAN Register Summary**

| Register Name | | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|---|
| 0x000F CANTXERR | R | TXERR7 | TXERR6 | TXERR5 | TXERR4 | TXERR3 | TXERR2 | TXERR1 | TXERR0 |
| | W | | | | | | | | |
| 0x0010–0x0013 CANIDAR0–3 | R | AC7 | AC6 | AC5 | AC4 | AC3 | AC2 | AC1 | AC0 |
| | W | | | | | | | | |
| 0x0014–0x0017 CANIDMRx | R | AM7 | AM6 | AM5 | AM4 | AM3 | AM2 | AM1 | AM0 |
| | W | | | | | | | | |
| 0x0018–0x001B CANIDAR4–7 | R | AC7 | AC6 | AC5 | AC4 | AC3 | AC2 | AC1 | AC0 |
| | W | | | | | | | | |
| 0x001C–0x001F CANIDMR4–7 | R | AM7 | AM6 | AM5 | AM4 | AM3 | AM2 | AM1 | AM0 |
| | W | | | | | | | | |
| 0x0020–0x002F CANRXFG | R | See Section 16.3.3, "Programmer's Model of Message Storage" | | | | | | | |
| | W | | | | | | | | |
| 0x0030–0x003F CANTXFG | R | See Section 16.3.3, "Programmer's Model of Message Storage" | | | | | | | |
| | W | | | | | | | | |

= Unimplemented or Reserved

**Figure 16-3. MSCAN Register Summary (continued)**

## 16.3.2 Register Descriptions

This section describes in detail all the registers and register bits in the MSCAN module. Each description includes a standard register diagram with an associated figure number. Details of register bit and field function follow the register diagrams, in bit order. All bits of all registers in this module are completely synchronous to internal clocks during a register read.

### 16.3.2.1 MSCAN Control Register 0 (CANCTL0)

The CANCTL0 register provides various control bits of the MSCAN module as described below.

Module Base + 0x0000 — Access: User read/write(1)

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | RXFRM | RXACT | CSWAI | SYNCH | TIME | WUPE | SLPRQ | INITRQ |
| W | | | | | | | | |
| Reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |

= Unimplemented

**Figure 16-4. MSCAN Control Register 0 (CANCTL0)**

1. Read: Anytime
   Write: Anytime when out of initialization mode; exceptions are read-only RXACT and SYNCH, RXFRM (which is set by the module only), and INITRQ (which is also writable in initialization mode)

**NOTE**

The CANCTL0 register, except WUPE, INITRQ, and SLPRQ, is held in the reset state when the initialization mode is active (INITRQ = 1 and INITAK = 1). This register is writable again as soon as the initialization mode is exited (INITRQ = 0 and INITAK = 0).

**Table 16-3. CANCTL0 Register Field Descriptions**

| Field | Description |
|---|---|
| 7<br>RXFRM | **Received Frame Flag** — This bit is read and clear only. It is set when a receiver has received a valid message correctly, independently of the filter configuration. After it is set, it remains set until cleared by software or reset. Clearing is done by writing a 1. Writing a 0 is ignored. This bit is not valid in loopback mode.<br>0 No valid message was received since last clearing this flag<br>1 A valid message was received since last clearing of this flag |
| 6<br>RXACT | **Receiver Active Status** — This read-only flag indicates the MSCAN is receiving a message(1). The flag is controlled by the receiver front end. This bit is not valid in loopback mode.<br>0 MSCAN is transmitting or idle<br>1 MSCAN is receiving a message (including when arbitration is lost) |
| 5<br>CSWAI(2) | **CAN Stops in Wait Mode** — Enabling this bit allows for lower power consumption in wait mode by disabling all the clocks at the CPU bus interface to the MSCAN module.<br>0 The module is not affected during wait mode<br>1 The module ceases to be clocked during wait mode |
| 4<br>SYNCH | **Synchronized Status** — This read-only flag indicates whether the MSCAN is synchronized to the CAN bus and able to participate in the communication process. It is set and cleared by the MSCAN.<br>0 MSCAN is not synchronized to the CAN bus<br>1 MSCAN is synchronized to the CAN bus |
| 3<br>TIME | **Timer Enable** — This bit activates an internal 16-bit wide free running timer which is clocked by the bit clock rate. If the timer is enabled, a 16-bit time stamp will be assigned to each transmitted/received message within the active TX/RX buffer. Right after the EOF of a valid message on the CAN bus, the time stamp is written to the highest bytes (0x000E, 0x000F) in the appropriate buffer (see Section 16.3.3, "Programmer's Model of Message Storage"). In loopback mode no receive timestamp is generated. The internal timer is reset (all bits set to 0) when disabled. This bit is held low in initialization mode.<br>0 Disable internal MSCAN timer<br>1 Enable internal MSCAN timer |
| 2<br>WUPE(3) | **Wake-Up Enable** — This configuration bit allows the MSCAN to restart from sleep mode or from power down mode (entered from sleep) when traffic on CAN is detected (see Section 16.4.5.5, "MSCAN Sleep Mode"). This bit must be configured before sleep mode entry for the selected function to take effect.<br>0 Wake-up disabled — The MSCAN ignores traffic on CAN<br>1 Wake-up enabled — The MSCAN is able to restart |

**Table 16-3. CANCTL0 Register Field Descriptions (continued)**

| Field | Description |
|---|---|
| 1<br>SLPRQ[4] | **Sleep Mode Request** — This bit requests the MSCAN to enter sleep mode, which is an internal power saving mode (see Section 16.4.5.5, "MSCAN Sleep Mode"). The sleep mode request is serviced when the CAN bus is idle, i.e., the module is not receiving a message and all transmit buffers are empty. The module indicates entry to sleep mode by setting SLPAK = 1 (see Section 16.3.2.2, "MSCAN Control Register 1 (CANCTL1)"). SLPRQ cannot be set while the WUPIF flag is set (see Section 16.3.2.5, "MSCAN Receiver Flag Register (CANRFLG)"). Sleep mode will be active until SLPRQ is cleared by the CPU or, depending on the setting of WUPE, the MSCAN detects activity on the CAN bus and clears SLPRQ itself.<br>0 Running — The MSCAN functions normally<br>1 Sleep mode request — The MSCAN enters sleep mode when CAN bus idle |
| 0<br>INITRQ[5],[6] | **Initialization Mode Request** — When this bit is set by the CPU, the MSCAN skips to initialization mode (see Section 16.4.4.5, "MSCAN Initialization Mode"). Any ongoing transmission or reception is aborted and synchronization to the CAN bus is lost. The module indicates entry to initialization mode by setting INITAK = 1 (Section 16.3.2.2, "MSCAN Control Register 1 (CANCTL1)").<br>The following registers enter their hard reset state and restore their default values: CANCTL0[7], CANRFLG[8], CANRIER[9], CANTFLG, CANTIER, CANTARQ, CANTAAK, and CANTBSEL.<br>The registers CANCTL1, CANBTR0, CANBTR1, CANIDAC, CANIDAR0-7, and CANIDMR0-7 can only be written by the CPU when the MSCAN is in initialization mode (INITRQ = 1 and INITAK = 1). The values of the error counters are not affected by initialization mode.<br>When this bit is cleared by the CPU, the MSCAN restarts and then tries to synchronize to the CAN bus. If the MSCAN is not in bus-off state, it synchronizes after 11 consecutive recessive bits on the CAN bus; if the MSCAN is in bus-off state, it continues to wait for 128 occurrences of 11 consecutive recessive bits.<br>Writing to other bits in CANCTL0, CANRFLG, CANRIER, CANTFLG, or CANTIER must be done only after initialization mode is exited, which is INITRQ = 0 and INITAK = 0.<br>0 Normal operation<br>1 MSCAN in initialization mode |

1. See the Bosch CAN 2.0A/B specification for a detailed definition of transmitter and receiver states.
2. In order to protect from accidentally violating the CAN protocol, TXCAN is immediately forced to a recessive state when the CPU enters wait (CSWAI = 1) or stop mode (see Section 16.4.5.2, "Operation in Wait Mode" and Section 16.4.5.3, "Operation in Stop Mode").
3. The CPU has to make sure that the WUPE register and the WUPIE wake-up interrupt enable register (see Section 16.3.2.6, "MSCAN Receiver Interrupt Enable Register (CANRIER)) is enabled, if the recovery mechanism from stop or wait is required.
4. The CPU cannot clear SLPRQ before the MSCAN has entered sleep mode (SLPRQ = 1 and SLPAK = 1).
5. The CPU cannot clear INITRQ before the MSCAN has entered initialization mode (INITRQ = 1 and INITAK = 1).
6. In order to protect from accidentally violating the CAN protocol, TXCAN is immediately forced to a recessive state when the initialization mode is requested by the CPU. Thus, the recommended procedure is to bring the MSCAN into sleep mode (SLPRQ = 1 and SLPAK = 1) before requesting initialization mode.
7. Not including WUPE, INITRQ, and SLPRQ.
8. TSTAT1 and TSTAT0 are not affected by initialization mode.
9. RSTAT1 and RSTAT0 are not affected by initialization mode.

### 16.3.2.2 MSCAN Control Register 1 (CANCTL1)

The CANCTL1 register provides various control bits and handshake status information of the MSCAN module as described below.

Module Base + 0x0001 Access: User read/write(1)

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | CANE | CLKSRC | LOOPB | LISTEN | BORM | WUPM | SLPAK | INITAK |
| W | | | | | | | | |
| Reset: | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 |

= Unimplemented

**Figure 16-5. MSCAN Control Register 1 (CANCTL1)**

1. Read: Anytime
   Write: Anytime in initialization mode (INITRQ = 1 and INITAK = 1), except CANE which is write once in normal and anytime in special system operation modes when the MSCAN is in initialization mode (INITRQ = 1 and INITAK = 1)

**Table 16-4. CANCTL1 Register Field Descriptions**

| Field | Description |
|---|---|
| 7<br>CANE | **MSCAN Enable**<br>0 MSCAN module is disabled<br>1 MSCAN module is enabled |
| 6<br>CLKSRC | **MSCAN Clock Source** — This bit defines the clock source for the MSCAN module (only for systems with a clock generation module; Section 16.4.3.2, "Clock System," and Section Figure 16-43., "MSCAN Clocking Scheme,").<br>0 MSCAN clock source is the oscillator clock<br>1 MSCAN clock source is the bus clock |
| 5<br>LOOPB | **Loopback Self Test Mode** — When this bit is set, the MSCAN performs an internal loopback which can be used for self test operation. The bit stream output of the transmitter is fed back to the receiver internally. The RXCAN input is ignored and the TXCAN output goes to the recessive state (logic 1). The MSCAN behaves as it does normally when transmitting and treats its own transmitted message as a message received from a remote node. In this state, the MSCAN ignores the bit sent during the ACK slot in the CAN frame acknowledge field to ensure proper reception of its own message. Both transmit and receive interrupts are generated.<br>0 Loopback self test disabled<br>1 Loopback self test enabled |
| 4<br>LISTEN | **Listen Only Mode** — This bit configures the MSCAN as a CAN bus monitor. When LISTEN is set, all valid CAN messages with matching ID are received, but no acknowledgement or error frames are sent out (see Section 16.4.4.4, "Listen-Only Mode"). In addition, the error counters are frozen. Listen only mode supports applications which require "hot plugging" or throughput analysis. The MSCAN is unable to transmit any messages when listen only mode is active.<br>0 Normal operation<br>1 Listen only mode activated |
| 3<br>BORM | **Bus-Off Recovery Mode** — This bit configures the bus-off state recovery mode of the MSCAN. Refer to Section 16.5.2, "Bus-Off Recovery," for details.<br>0 Automatic bus-off recovery (see Bosch CAN 2.0A/B protocol specification)<br>1 Bus-off recovery upon user request |
| 2<br>WUPM | **Wake-Up Mode** — If WUPE in CANCTL0 is enabled, this bit defines whether the integrated low-pass filter is applied to protect the MSCAN from spurious wake-up (see Section 16.4.5.5, "MSCAN Sleep Mode").<br>0 MSCAN wakes up on any dominant level on the CAN bus<br>1 MSCAN wakes up only in case of a dominant pulse on the CAN bus that has a length of $T_{wup}$ |

**Table 16-4. CANCTL1 Register Field Descriptions (continued)**

| Field | Description |
|---|---|
| 1<br>SLPAK | **Sleep Mode Acknowledge** — This flag indicates whether the MSCAN module has entered sleep mode (see Section 16.4.5.5, "MSCAN Sleep Mode"). It is used as a handshake flag for the SLPRQ sleep mode request. Sleep mode is active when SLPRQ = 1 and SLPAK = 1. Depending on the setting of WUPE, the MSCAN will clear the flag if it detects activity on the CAN bus while in sleep mode.<br>0 Running — The MSCAN operates normally<br>1 Sleep mode active — The MSCAN has entered sleep mode |
| 0<br>INITAK | **Initialization Mode Acknowledge** — This flag indicates whether the MSCAN module is in initialization mode (see Section 16.4.4.5, "MSCAN Initialization Mode"). It is used as a handshake flag for the INITRQ initialization mode request. Initialization mode is active when INITRQ = 1 and INITAK = 1. The registers CANCTL1, CANBTR0, CANBTR1, CANIDAC, CANIDAR0–CANIDAR7, and CANIDMR0–CANIDMR7 can be written only by the CPU when the MSCAN is in initialization mode.<br>0 Running — The MSCAN operates normally<br>1 Initialization mode active — The MSCAN has entered initialization mode |

### 16.3.2.3 MSCAN Bus Timing Register 0 (CANBTR0)

The CANBTR0 register configures various CAN bus timing parameters of the MSCAN module.

Module Base + 0x0002 Access: User read/write(1)

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | SJW1 | SJW0 | BRP5 | BRP4 | BRP3 | BRP2 | BRP1 | BRP0 |
| Reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 16-6. MSCAN Bus Timing Register 0 (CANBTR0)**

1. Read: Anytime
   Write: Anytime in initialization mode (INITRQ = 1 and INITAK = 1)

**Table 16-5. CANBTR0 Register Field Descriptions**

| Field | Description |
|---|---|
| 7-6<br>SJW[1:0] | **Synchronization Jump Width** — The synchronization jump width defines the maximum number of time quanta (Tq) clock cycles a bit can be shortened or lengthened to achieve resynchronization to data transitions on the CAN bus (see Table 16-6). |
| 5-0<br>BRP[5:0] | **Baud Rate Prescaler** — These bits determine the time quanta (Tq) clock which is used to build up the bit timing (see Table 16-7). |

**Table 16-6. Synchronization Jump Width**

| SJW1 | SJW0 | Synchronization Jump Width |
|---|---|---|
| 0 | 0 | 1 Tq clock cycle |
| 0 | 1 | 2 Tq clock cycles |
| 1 | 0 | 3 Tq clock cycles |
| 1 | 1 | 4 Tq clock cycles |

**Table 16-7. Baud Rate Prescaler**

| BRP5 | BRP4 | BRP3 | BRP2 | BRP1 | BRP0 | Prescaler value (P) |
|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| 0 | 0 | 0 | 0 | 0 | 1 | 2 |
| 0 | 0 | 0 | 0 | 1 | 0 | 3 |
| 0 | 0 | 0 | 0 | 1 | 1 | 4 |
| : | : | : | : | : | : | : |
| 1 | 1 | 1 | 1 | 1 | 1 | 64 |

### 16.3.2.4 MSCAN Bus Timing Register 1 (CANBTR1)

The CANBTR1 register configures various CAN bus timing parameters of the MSCAN module.

Module Base + 0x0003 Access: User read/write(1)

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | SAMP | TSEG22 | TSEG21 | TSEG20 | TSEG13 | TSEG12 | TSEG11 | TSEG10 |
| Reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 16-7. MSCAN Bus Timing Register 1 (CANBTR1)**

1. Read: Anytime
   Write: Anytime in initialization mode (INITRQ = 1 and INITAK = 1)

**Table 16-8. CANBTR1 Register Field Descriptions**

| Field | Description |
|---|---|
| 7<br>SAMP | **Sampling** — This bit determines the number of CAN bus samples taken per bit time.<br>0 One sample per bit.<br>1 Three samples per bit(1).<br>If SAMP = 0, the resulting bit value is equal to the value of the single bit positioned at the sample point. If SAMP = 1, the resulting bit value is determined by using majority rule on the three total samples. For higher bit rates, it is recommended that only one sample is taken per bit time (SAMP = 0). |
| 6-4<br>TSEG2[2:0] | **Time Segment 2** — Time segments within the bit time fix the number of clock cycles per bit time and the location of the sample point (see Figure 16-44). Time segment 2 (TSEG2) values are programmable as shown in Table 16-9. |
| 3-0<br>TSEG1[3:0] | **Time Segment 1** — Time segments within the bit time fix the number of clock cycles per bit time and the location of the sample point (see Figure 16-44). Time segment 1 (TSEG1) values are programmable as shown in Table 16-10. |

1. In this case, PHASE_SEG1 must be at least 2 time quanta (Tq).

**Table 16-9. Time Segment 2 Values**

| TSEG22 | TSEG21 | TSEG20 | Time Segment 2 |
|---|---|---|---|
| 0 | 0 | 0 | 1 Tq clock cycle(1) |
| 0 | 0 | 1 | 2 Tq clock cycles |
| : | : | : | : |
| 1 | 1 | 0 | 7 Tq clock cycles |
| 1 | 1 | 1 | 8 Tq clock cycles |

1. This setting is not valid. Please refer to Table 16-37 for valid settings.

**Table 16-10. Time Segment 1 Values**

| TSEG13 | TSEG12 | TSEG11 | TSEG10 | Time segment 1 |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 1 Tq clock cycle(1) |
| 0 | 0 | 0 | 1 | 2 Tq clock cycles[1] |
| 0 | 0 | 1 | 0 | 3 Tq clock cycles[1] |
| 0 | 0 | 1 | 1 | 4 Tq clock cycles |
| : | : | : | : | : |
| 1 | 1 | 1 | 0 | 15 Tq clock cycles |
| 1 | 1 | 1 | 1 | 16 Tq clock cycles |

1. This setting is not valid. Please refer to Table 16-37 for valid settings.

The bit time is determined by the oscillator frequency, the baud rate prescaler, and the number of time quanta (Tq) clock cycles per bit (as shown in Table 16-9 and Table 16-10).

***Eqn. 16-1***

$$\text{Bit Time} = \frac{(\text{Prescaler value})}{f_{\text{CANCLK}}} \bullet (1 + \text{TimeSegment1} + \text{TimeSegment2})$$

### 16.3.2.5 MSCAN Receiver Flag Register (CANRFLG)

A flag can be cleared only by software (writing a 1 to the corresponding bit position) when the condition which caused the setting is no longer valid. Every flag has an associated interrupt enable bit in the CANRIER register.

Module Base + 0x0004 — Access: User read/write(1)

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | WUPIF | CSCIF | RSTAT1 | RSTAT0 | TSTAT1 | TSTAT0 | OVRIF | RXF |
| W | | | | | | | | |
| Reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented

**Figure 16-8. MSCAN Receiver Flag Register (CANRFLG)**

1. Read: Anytime
   Write: Anytime when not in initialization mode, except RSTAT[1:0] and TSTAT[1:0] flags which are read-only; write of 1 clears flag; write of 0 is ignored

**NOTE**

The CANRFLG register is held in the reset state[1] when the initialization mode is active (INITRQ = 1 and INITAK = 1). This register is writable again as soon as the initialization mode is exited (INITRQ = 0 and INITAK = 0).

**Table 16-11. CANRFLG Register Field Descriptions**

| Field | Description |
|---|---|
| 7<br>WUPIF | **Wake-Up Interrupt Flag** — If the MSCAN detects CAN bus activity while in sleep mode (see Section 16.4.5.5, "MSCAN Sleep Mode,") and WUPE = 1 in CANTCTL0 (see Section 16.3.2.1, "MSCAN Control Register 0 (CANCTL0)"), the module will set WUPIF. If not masked, a wake-up interrupt is pending while this flag is set.<br>0 No wake-up activity observed while in sleep mode<br>1 MSCAN detected activity on the CAN bus and requested wake-up |
| 6<br>CSCIF | **CAN Status Change Interrupt Flag** — This flag is set when the MSCAN changes its current CAN bus status due to the actual value of the transmit error counter (TEC) and the receive error counter (REC). An additional 4-bit (RSTAT[1:0], TSTAT[1:0]) status register, which is split into separate sections for TEC/REC, informs the system on the actual CAN bus status (see Section 16.3.2.6, "MSCAN Receiver Interrupt Enable Register (CANRIER)"). If not masked, an error interrupt is pending while this flag is set. CSCIF provides a blocking interrupt. That guarantees that the receiver/transmitter status bits (RSTAT/TSTAT) are only updated when no CAN status change interrupt is pending. If the TECs/RECs change their current value after the CSCIF is asserted, which would cause an additional state change in the RSTAT/TSTAT bits, these bits keep their status until the current CSCIF interrupt is cleared again.<br>0 No change in CAN bus status occurred since last interrupt<br>1 MSCAN changed current CAN bus status |
| 5-4<br>RSTAT[1:0] | **Receiver Status Bits** — The values of the error counters control the actual CAN bus status of the MSCAN. As soon as the status change interrupt flag (CSCIF) is set, these bits indicate the appropriate receiver related CAN bus status of the MSCAN. The coding for the bits RSTAT1, RSTAT0 is:<br>00 RxOK: 0 ≤ receive error counter < 96<br>01 RxWRN: 96 ≤ receive error counter < 128<br>10 RxERR: 128 ≤ receive error counter<br>*11 Bus-off[1]: 256 ≤ transmit error counter* |
| 3-2<br>TSTAT[1:0] | **Transmitter Status Bits** — The values of the error counters control the actual CAN bus status of the MSCAN. As soon as the status change interrupt flag (CSCIF) is set, these bits indicate the appropriate transmitter related CAN bus status of the MSCAN. The coding for the bits TSTAT1, TSTAT0 is:<br>00 TxOK: 0 ≤ transmit error counter < 96<br>01 TxWRN: 96 ≤ transmit error counter < 128<br>10 TxERR: 128 ≤ transmit error counter < 256<br>11 Bus-Off: 256 ≤ transmit error counter |

1. The RSTAT[1:0], TSTAT[1:0] bits are not affected by initialization mode.

**Table 16-11. CANRFLG Register Field Descriptions (continued)**

| Field | Description |
|---|---|
| 1<br>OVRIF | **Overrun Interrupt Flag** — This flag is set when a data overrun condition occurs. If not masked, an error interrupt is pending while this flag is set.<br>0 No data overrun condition<br>1 A data overrun detected |
| 0<br>RXF(2) | **Receive Buffer Full Flag** — RXF is set by the MSCAN when a new message is shifted in the receiver FIFO. This flag indicates whether the shifted buffer is loaded with a correctly received message (matching identifier, matching cyclic redundancy code (CRC) and no other errors detected). After the CPU has read that message from the RxFG buffer in the receiver FIFO, the RXF flag must be cleared to release the buffer. A set RXF flag prohibits the shifting of the next FIFO entry into the foreground buffer (RxFG). If not masked, a receive interrupt is pending while this flag is set.<br>0 No new message available within the RxFG<br>1 The receiver FIFO is not empty. A new message is available in the RxFG |

1. Redundant Information for the most critical CAN bus status which is "bus-off". This only occurs if the Tx error counter exceeds a number of 255 errors. Bus-off affects the receiver state. As soon as the transmitter leaves its bus-off state the receiver state skips to RxOK too. Refer also to TSTAT[1:0] coding in this register.
2. To ensure data integrity, do not read the receive buffer registers while the RXF flag is cleared. For MCUs with dual CPUs, reading the receive buffer registers while the RXF flag is cleared may result in a CPU fault condition.

### 16.3.2.6 MSCAN Receiver Interrupt Enable Register (CANRIER)

This register contains the interrupt enable bits for the interrupt flags described in the CANRFLG register.

Module Base + 0x0005 Access: User read/write(1)

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | WUPIE | CSCIE | RSTATE1 | RSTATE0 | TSTATE1 | TSTATE0 | OVRIE | RXFIE |
| Reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 16-9. MSCAN Receiver Interrupt Enable Register (CANRIER)**

1. Read: Anytime
   Write: Anytime when not in initialization mode

**NOTE**

The CANRIER register is held in the reset state when the initialization mode is active (INITRQ=1 and INITAK=1). This register is writable when not in initialization mode (INITRQ=0 and INITAK=0).

The RSTATE[1:0], TSTATE[1:0] bits are not affected by initialization mode.

**Table 16-12. CANRIER Register Field Descriptions**

| Field | Description |
|---|---|
| 7<br>WUPIE[1] | **Wake-Up Interrupt Enable**<br>0 No interrupt request is generated from this event.<br>1 A wake-up event causes a Wake-Up interrupt request. |
| 6<br>CSCIE | **CAN Status Change Interrupt Enable**<br>0 No interrupt request is generated from this event.<br>1 A CAN Status Change event causes an error interrupt request. |
| 5-4<br>RSTATE[1:0] | **Receiver Status Change Enable** — These RSTAT enable bits control the sensitivity level in which receiver state changes are causing CSCIF interrupts. Independent of the chosen sensitivity level the RSTAT flags continue to indicate the actual receiver state and are only updated if no CSCIF interrupt is pending.<br>00 Do not generate any CSCIF interrupt caused by receiver state changes.<br>01 Generate CSCIF interrupt only if the receiver enters or leaves "bus-off" state. Discard other receiver state changes for generating CSCIF interrupt.<br>10 Generate CSCIF interrupt only if the receiver enters or leaves "RxErr" or "bus-off"[2] state. Discard other receiver state changes for generating CSCIF interrupt.<br>11 Generate CSCIF interrupt on all state changes. |
| 3-2<br>TSTATE[1:0] | **Transmitter Status Change Enable** — These TSTAT enable bits control the sensitivity level in which transmitter state changes are causing CSCIF interrupts. Independent of the chosen sensitivity level, the TSTAT flags continue to indicate the actual transmitter state and are only updated if no CSCIF interrupt is pending.<br>00 Do not generate any CSCIF interrupt caused by transmitter state changes.<br>01 Generate CSCIF interrupt only if the transmitter enters or leaves "bus-off" state. Discard other transmitter state changes for generating CSCIF interrupt.<br>10 Generate CSCIF interrupt only if the transmitter enters or leaves "TxErr" or "bus-off" state. Discard other transmitter state changes for generating CSCIF interrupt.<br>11 Generate CSCIF interrupt on all state changes. |
| 1<br>OVRIE | **Overrun Interrupt Enable**<br>0 No interrupt request is generated from this event.<br>1 An overrun event causes an error interrupt request. |
| 0<br>RXFIE | **Receiver Full Interrupt Enable**<br>0 No interrupt request is generated from this event.<br>1 A receive buffer full (successful message reception) event causes a receiver interrupt request. |

1. WUPIE and WUPE (see Section 16.3.2.1, "MSCAN Control Register 0 (CANCTL0)") must both be enabled if the recovery mechanism from stop or wait is required.
2. Bus-off state is only defined for transmitters by the CAN standard (see Bosch CAN 2.0A/B protocol specification). Because the only possible state change for the transmitter from bus-off to TxOK also forces the receiver to skip its current state to RxOK, the coding of the RXSTAT[1:0] flags define an additional bus-off state for the receiver (see Section 16.3.2.5, "MSCAN Receiver Flag Register (CANRFLG)").

### 16.3.2.7 MSCAN Transmitter Flag Register (CANTFLG)

The transmit buffer empty flags each have an associated interrupt enable bit in the CANTIER register.

Module Base + 0x0006 Access: User read/write(1)

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | 0 | 0 | 0 | TXE2 | TXE1 | TXE0 |
| W | | | | | | | | |
| Reset: | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 |

= Unimplemented

**Figure 16-10. MSCAN Transmitter Flag Register (CANTFLG)**

1. Read: Anytime
   Write: Anytime when not in initialization mode; write of 1 clears flag, write of 0 is ignored

**NOTE**

The CANTFLG register is held in the reset state when the initialization mode is active (INITRQ = 1 and INITAK = 1). This register is writable when not in initialization mode (INITRQ = 0 and INITAK = 0).

**Table 16-13. CANTFLG Register Field Descriptions**

| Field | Description |
|---|---|
| 2-0<br>TXE[2:0] | **Transmitter Buffer Empty** — This flag indicates that the associated transmit message buffer is empty, and thus not scheduled for transmission. The CPU must clear the flag after a message is set up in the transmit buffer and is due for transmission. The MSCAN sets the flag after the message is sent successfully. The flag is also set by the MSCAN when the transmission request is successfully aborted due to a pending abort request (see Section 16.3.2.9, "MSCAN Transmitter Message Abort Request Register (CANTARQ)"). If not masked, a transmit interrupt is pending while this flag is set.<br>Clearing a TXEx flag also clears the corresponding ABTAKx (see Section 16.3.2.10, "MSCAN Transmitter Message Abort Acknowledge Register (CANTAAK)"). When a TXEx flag is set, the corresponding ABTRQx bit is cleared (see Section 16.3.2.9, "MSCAN Transmitter Message Abort Request Register (CANTARQ)").<br>When listen-mode is active (see Section 16.3.2.2, "MSCAN Control Register 1 (CANCTL1)") the TXEx flags cannot be cleared and no transmission is started.<br>Read and write accesses to the transmit buffer will be blocked, if the corresponding TXEx bit is cleared (TXEx = 0) and the buffer is scheduled for transmission.<br>0 The associated message buffer is full (loaded with a message due for transmission)<br>1 The associated message buffer is empty (not scheduled) |

## 16.3.2.8 MSCAN Transmitter Interrupt Enable Register (CANTIER)

This register contains the interrupt enable bits for the transmit buffer empty interrupt flags.

Module Base + 0x0007 Access: User read/write(1)

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | 0 | 0 | 0 | TXEIE2 | TXEIE1 | TXEIE0 |
| W | | | | | | | | |
| Reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented

**Figure 16-11. MSCAN Transmitter Interrupt Enable Register (CANTIER)**

1. Read: Anytime
   Write: Anytime when not in initialization mode

**NOTE**

The CANTIER register is held in the reset state when the initialization mode is active (INITRQ = 1 and INITAK = 1). This register is writable when not in initialization mode (INITRQ = 0 and INITAK = 0).

**Table 16-14. CANTIER Register Field Descriptions**

| Field | Description |
|---|---|
| 2-0<br>TXEIE[2:0] | **Transmitter Empty Interrupt Enable**<br>0 No interrupt request is generated from this event.<br>1 A transmitter empty (transmit buffer available for transmission) event causes a transmitter empty interrupt request. |

### 16.3.2.9 MSCAN Transmitter Message Abort Request Register (CANTARQ)

The CANTARQ register allows abort request of queued messages as described below.

Module Base + 0x0008 Access: User read/write(1)

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | 0 | 0 | 0 | ABTRQ2 | ABTRQ1 | ABTRQ0 |
| W | | | | | | | | |
| Reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented

**Figure 16-12. MSCAN Transmitter Message Abort Request Register (CANTARQ)**

1. Read: Anytime
   Write: Anytime when not in initialization mode

**NOTE**

The CANTARQ register is held in the reset state when the initialization mode is active (INITRQ = 1 and INITAK = 1). This register is writable when not in initialization mode (INITRQ = 0 and INITAK = 0).

**Table 16-15. CANTARQ Register Field Descriptions**

| Field | Description |
|---|---|
| 2-0<br>ABTRQ[2:0] | **Abort Request** — The CPU sets the ABTRQx bit to request that a scheduled message buffer (TXEx = 0) be aborted. The MSCAN grants the request if the message has not already started transmission, or if the transmission is not successful (lost arbitration or error). When a message is aborted, the associated TXE (see Section 16.3.2.7, "MSCAN Transmitter Flag Register (CANTFLG)") and abort acknowledge flags (ABTAK, see Section 16.3.2.10, "MSCAN Transmitter Message Abort Acknowledge Register (CANTAAK)") are set and a transmit interrupt occurs if enabled. The CPU cannot reset ABTRQx. ABTRQx is reset whenever the associated TXE flag is set.<br>0 No abort request<br>1 Abort request pending |

### 16.3.2.10 MSCAN Transmitter Message Abort Acknowledge Register (CANTAAK)

The CANTAAK register indicates the successful abort of a queued message, if requested by the appropriate bits in the CANTARQ register.

Module Base + 0x0009 Access: User read/write(1)

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | 0 | 0 | 0 | ABTAK2 | ABTAK1 | ABTAK0 |
| W | | | | | | | | |
| Reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented

**Figure 16-13. MSCAN Transmitter Message Abort Acknowledge Register (CANTAAK)**

1. Read: Anytime
   Write: Unimplemented

**NOTE**

The CANTAAK register is held in the reset state when the initialization mode is active (INITRQ = 1 and INITAK = 1).

**Table 16-16. CANTAAK Register Field Descriptions**

| Field | Description |
|---|---|
| 2-0<br>ABTAK[2:0] | **Abort Acknowledge** — This flag acknowledges that a message was aborted due to a pending abort request from the CPU. After a particular message buffer is flagged empty, this flag can be used by the application software to identify whether the message was aborted successfully or was sent anyway. The ABTAKx flag is cleared whenever the corresponding TXE flag is cleared.<br>0 The message was not aborted.<br>1 The message was aborted. |

### 16.3.2.11 MSCAN Transmit Buffer Selection Register (CANTBSEL)

The CANTBSEL register allows the selection of the actual transmit message buffer, which then will be accessible in the CANTXFG register space.

Module Base + 0x000A Access: User read/write(1)

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | 0 | 0 | 0 | TX2 | TX1 | TX0 |
| W | | | | | | | | |
| Reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented

**Figure 16-14. MSCAN Transmit Buffer Selection Register (CANTBSEL)**

1. Read: Find the lowest ordered bit set to 1, all other bits will be read as 0
   Write: Anytime when not in initialization mode

**NOTE**

The CANTBSEL register is held in the reset state when the initialization mode is active (INITRQ = 1 and INITAK=1). This register is writable when not in initialization mode (INITRQ = 0 and INITAK = 0).

**Table 16-17. CANTBSEL Register Field Descriptions**

| Field | Description |
|---|---|
| 2-0<br>TX[2:0] | **Transmit Buffer Select** — The lowest numbered bit places the respective transmit buffer in the CANTXFG register space (e.g., TX1 = 1 and TX0 = 1 selects transmit buffer TX0; TX1 = 1 and TX0 = 0 selects transmit buffer TX1). Read and write accesses to the selected transmit buffer will be blocked, if the corresponding TXEx bit is cleared and the buffer is scheduled for transmission (see Section 16.3.2.7, "MSCAN Transmitter Flag Register (CANTFLG)").<br>0 The associated message buffer is deselected<br>1 The associated message buffer is selected, if lowest numbered bit |

The following gives a short programming example of the usage of the CANTBSEL register:

To get the next available transmit buffer, application software must read the CANTFLG register and write this value back into the CANTBSEL register. In this example Tx buffers TX1 and TX2 are available. The value read from CANTFLG is therefore 0b0000_0110. When writing this value back to CANTBSEL, the Tx buffer TX1 is selected in the CANTXFG because the lowest numbered bit set to 1 is at bit position 1. Reading back this value out of CANTBSEL results in 0b0000_0010, because only the lowest numbered bit position set to 1 is presented. This mechanism eases the application software's selection of the next available Tx buffer.

- LDAA CANTFLG; value read is 0b0000_0110
- STAA CANTBSEL; value written is 0b0000_0110
- LDAA CANTBSEL; value read is 0b0000_0010

If all transmit message buffers are deselected, no accesses are allowed to the CANTXFG registers.

### 16.3.2.12 MSCAN Identifier Acceptance Control Register (CANIDAC)

The CANIDAC register is used for identifier acceptance control as described below.

Module Base + 0x000B — Access: User read/write(1)

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | IDAM1 | IDAM0 | 0 | IDHIT2 | IDHIT1 | IDHIT0 |
| W | | | | | | | | |
| Reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented

**Figure 16-15. MSCAN Identifier Acceptance Control Register (CANIDAC)**

1. Read: Anytime
   Write: Anytime in initialization mode (INITRQ = 1 and INITAK = 1), except bits IDHITx, which are read-only

**Table 16-18. CANIDAC Register Field Descriptions**

| Field | Description |
|---|---|
| 5-4<br>IDAM[1:0] | **Identifier Acceptance Mode** — The CPU sets these flags to define the identifier acceptance filter organization (see Section 16.4.3, "Identifier Acceptance Filter"). Table 16-19 summarizes the different settings. In filter closed mode, no message is accepted such that the foreground buffer is never reloaded. |
| 2-0<br>IDHIT[2:0] | Identifier Acceptance Hit Indicator — The MSCAN sets these flags to indicate an identifier acceptance hit (see Section 16.4.3, "Identifier Acceptance Filter"). Table 16-20 summarizes the different settings. |

**Table 16-19. Identifier Acceptance Mode Settings**

| IDAM1 | IDAM0 | Identifier Acceptance Mode |
|---|---|---|
| 0 | 0 | Two 32-bit acceptance filters |
| 0 | 1 | Four 16-bit acceptance filters |
| 1 | 0 | Eight 8-bit acceptance filters |
| 1 | 1 | Filter closed |

**Table 16-20. Identifier Acceptance Hit Indication**

| IDHIT2 | IDHIT1 | IDHIT0 | Identifier Acceptance Hit |
|---|---|---|---|
| 0 | 0 | 0 | Filter 0 hit |
| 0 | 0 | 1 | Filter 1 hit |
| 0 | 1 | 0 | Filter 2 hit |
| 0 | 1 | 1 | Filter 3 hit |
| 1 | 0 | 0 | Filter 4 hit |
| 1 | 0 | 1 | Filter 5 hit |
| 1 | 1 | 0 | Filter 6 hit |
| 1 | 1 | 1 | Filter 7 hit |

The IDHITx indicators are always related to the message in the foreground buffer (RxFG). When a message gets shifted into the foreground buffer of the receiver FIFO the indicators are updated as well.

### 16.3.2.13 MSCAN Reserved Register

This register is reserved for factory testing of the MSCAN module and is not available in normal system operating modes.

Module Base + 0x000C — Access: User read/write(1)

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| W | | | | | | | | |
| Reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented

**Figure 16-16. MSCAN Reserved Register**

1. Read: Always reads zero in normal system operation modes
   Write: Unimplemented in normal system operation modes

**NOTE**

Writing to this register when in special system operating modes can alter the MSCAN functionality.

### 16.3.2.14 MSCAN Miscellaneous Register (CANMISC)

This register provides additional features.

Module Base + 0x000D — Access: User read/write(1)

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | BOHOLD |
| W | | | | | | | | |
| Reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented

**Figure 16-17. MSCAN Miscellaneous Register (CANMISC)**

1. Read: Anytime
   Write: Anytime; write of '1' clears flag; write of '0' ignored

**Table 16-21. CANMISC Register Field Descriptions**

| Field | Description |
|---|---|
| 0<br>BOHOLD | **Bus-off State Hold Until User Request** — If BORM is set in MSCAN Control Register 1 (CANCTL1), this bit indicates whether the module has entered the bus-off state. Clearing this bit requests the recovery from bus-off. Refer to Section 16.5.2, "Bus-Off Recovery," for details.<br>0 Module is not bus-off or recovery has been requested by user in bus-off state<br>1 Module is bus-off and holds this state until user request |

### 16.3.2.15 MSCAN Receive Error Counter (CANRXERR)

This register reflects the status of the MSCAN receive error counter.

Module Base + 0x000E — Access: User read/write(1)

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | RXERR7 | RXERR6 | RXERR5 | RXERR4 | RXERR3 | RXERR2 | RXERR1 | RXERR0 |
| W | | | | | | | | |
| Reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented

**Figure 16-18. MSCAN Receive Error Counter (CANRXERR)**

1. Read: Only when in sleep mode (SLPRQ = 1 and SLPAK = 1) or initialization mode (INITRQ = 1 and INITAK = 1)
   Write: Unimplemented

**NOTE**

Reading this register when in any other mode other than sleep or initialization mode may return an incorrect value. For MCUs with dual CPUs, this may result in a CPU fault condition.

### 16.3.2.16 MSCAN Transmit Error Counter (CANTXERR)

This register reflects the status of the MSCAN transmit error counter.

Module Base + 0x000F — Access: User read/write(1)

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | TXERR7 | TXERR6 | TXERR5 | TXERR4 | TXERR3 | TXERR2 | TXERR1 | TXERR0 |
| W | | | | | | | | |
| Reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented

**Figure 16-19. MSCAN Transmit Error Counter (CANTXERR)**

1. Read: Only when in sleep mode (SLPRQ = 1 and SLPAK = 1) or initialization mode (INITRQ = 1 and INITAK = 1)
   Write: Unimplemented

**NOTE**

Reading this register when in any other mode other than sleep or initialization mode, may return an incorrect value. For MCUs with dual CPUs, this may result in a CPU fault condition.

### 16.3.2.17 MSCAN Identifier Acceptance Registers (CANIDAR0-7)

On reception, each message is written into the background receive buffer. The CPU is only signalled to read the message if it passes the criteria in the identifier acceptance and identifier mask registers (accepted); otherwise, the message is overwritten by the next message (dropped).

The acceptance registers of the MSCAN are applied on the IDR0–IDR3 registers (see Section 16.3.3.1, "Identifier Registers (IDR0–IDR3)") of incoming messages in a bit by bit manner (see Section 16.4.3, "Identifier Acceptance Filter").

For extended identifiers, all four acceptance and mask registers are applied. For standard identifiers, only the first two (CANIDAR0/1, CANIDMR0/1) are applied.

Module Base + 0x0010 to Module Base + 0x0013 Access: User read/write(1)

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | AC7 | AC6 | AC5 | AC4 | AC3 | AC2 | AC1 | AC0 |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 16-20. MSCAN Identifier Acceptance Registers (First Bank) — CANIDAR0–CANIDAR3**

1. Read: Anytime
   Write: Anytime in initialization mode (INITRQ = 1 and INITAK = 1)

**Table 16-22. CANIDAR0–CANIDAR3 Register Field Descriptions**

| Field | Description |
|---|---|
| 7-0<br>AC[7:0] | **Acceptance Code Bits** — AC[7:0] comprise a user-defined sequence of bits with which the corresponding bits of the related identifier register (IDRn) of the receive message buffer are compared. The result of this comparison is then masked with the corresponding identifier mask register. |

Module Base + 0x0018 to Module Base + 0x001B Access: User read/write(1)

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | AC7 | AC6 | AC5 | AC4 | AC3 | AC2 | AC1 | AC0 |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 16-21. MSCAN Identifier Acceptance Registers (Second Bank) — CANIDAR4–CANIDAR7**

1. Read: Anytime
   Write: Anytime in initialization mode (INITRQ = 1 and INITAK = 1)

**Table 16-23. CANIDAR4–CANIDAR7 Register Field Descriptions**

| Field | Description |
|---|---|
| 7-0<br>AC[7:0] | **Acceptance Code Bits** — AC[7:0] comprise a user-defined sequence of bits with which the corresponding bits of the related identifier register (IDRn) of the receive message buffer are compared. The result of this comparison is then masked with the corresponding identifier mask register. |

### 16.3.2.18 MSCAN Identifier Mask Registers (CANIDMR0–CANIDMR7)

The identifier mask register specifies which of the corresponding bits in the identifier acceptance register are relevant for acceptance filtering. To receive standard identifiers in 32 bit filter mode, it is required to program the last three bits (AM[2:0]) in the mask registers CANIDMR1 and CANIDMR5 to "don't care." To receive standard identifiers in 16 bit filter mode, it is required to program the last three bits (AM[2:0]) in the mask registers CANIDMR1, CANIDMR3, CANIDMR5, and CANIDMR7 to "don't care."

Module Base + 0x0014 to Module Base + 0x0017 Access: User read/write(1)

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | AM7 | AM6 | AM5 | AM4 | AM3 | AM2 | AM1 | AM0 |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 16-22. MSCAN Identifier Mask Registers (First Bank) — CANIDMR0–CANIDMR3**

1. Read: Anytime
   Write: Anytime in initialization mode (INITRQ = 1 and INITAK = 1)

**Table 16-24. CANIDMR0–CANIDMR3 Register Field Descriptions**

| Field | Description |
|---|---|
| 7-0<br>AM[7:0] | **Acceptance Mask Bits** — If a particular bit in this register is cleared, this indicates that the corresponding bit in the identifier acceptance register must be the same as its identifier bit before a match is detected. The message is accepted if all such bits match. If a bit is set, it indicates that the state of the corresponding bit in the identifier acceptance register does not affect whether or not the message is accepted.<br>0 Match corresponding acceptance code register and identifier bits<br>1 Ignore corresponding acceptance code register bit |

Module Base + 0x001C to Module Base + 0x001F Access: User read/write(1)

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | AM7 | AM6 | AM5 | AM4 | AM3 | AM2 | AM1 | AM0 |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 16-23. MSCAN Identifier Mask Registers (Second Bank) — CANIDMR4–CANIDMR7**

1. Read: Anytime
Write: Anytime in initialization mode (INITRQ = 1 and INITAK = 1)

**Table 16-25. CANIDMR4–CANIDMR7 Register Field Descriptions**

| Field | Description |
|---|---|
| 7-0<br>AM[7:0] | **Acceptance Mask Bits** — If a particular bit in this register is cleared, this indicates that the corresponding bit in the identifier acceptance register must be the same as its identifier bit before a match is detected. The message is accepted if all such bits match. If a bit is set, it indicates that the state of the corresponding bit in the identifier acceptance register does not affect whether or not the message is accepted.<br>0 Match corresponding acceptance code register and identifier bits<br>1 Ignore corresponding acceptance code register bit |

## 16.3.3 Programmer's Model of Message Storage

The following section details the organization of the receive and transmit message buffers and the associated control registers.

To simplify the programmer interface, the receive and transmit message buffers have the same outline. Each message buffer allocates 16 bytes in the memory map containing a 13 byte data structure.

An additional transmit buffer priority register (TBPR) is defined for the transmit buffers. Within the last two bytes of this memory map, the MSCAN stores a special 16-bit time stamp, which is sampled from an internal timer after successful transmission or reception of a message. This feature is only available for transmit and receiver buffers, if the TIME bit is set (see Section 16.3.2.1, "MSCAN Control Register 0 (CANCTL0)").

The time stamp register is written by the MSCAN. The CPU can only read these registers.

**Table 16-26. Message Buffer Organization**

| Offset Address | Register | Access |
|---|---|---|
| 0x00X0 | IDR0 — Identifier Register 0 | R/W |
| 0x00X1 | IDR1 — Identifier Register 1 | R/W |
| 0x00X2 | IDR2 — Identifier Register 2 | R/W |
| 0x00X3 | IDR3 — Identifier Register 3 | R/W |
| 0x00X4 | DSR0 — Data Segment Register 0 | R/W |
| 0x00X5 | DSR1 — Data Segment Register 1 | R/W |
| 0x00X6 | DSR2 — Data Segment Register 2 | R/W |
| 0x00X7 | DSR3 — Data Segment Register 3 | R/W |
| 0x00X8 | DSR4 — Data Segment Register 4 | R/W |
| 0x00X9 | DSR5 — Data Segment Register 5 | R/W |
| 0x00XA | DSR6 — Data Segment Register 6 | R/W |
| 0x00XB | DSR7 — Data Segment Register 7 | R/W |
| 0x00XC | DLR — Data Length Register | R/W |
| 0x00XD | TBPR — Transmit Buffer Priority Register(1) | R/W |
| 0x00XE | TSRH — Time Stamp Register (High Byte) | R |
| 0x00XF | TSRL — Time Stamp Register (Low Byte) | R |

1. Not applicable for receive buffers

Figure 16-24 shows the common 13-byte data structure of receive and transmit buffers for extended identifiers. The mapping of standard identifiers into the IDR registers is shown in Figure 16-25.

All bits of the receive and transmit buffers are 'x' out of reset because of RAM-based implementation[1]. All reserved or unused bits of the receive and transmit buffers always read 'x'.

1. Exception: The transmit buffer priority registers are 0 out of reset.

**Figure 16-24. Receive/Transmit Message Buffer — Extended Identifier Mapping**

| Register Name | | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit0 |
|---|---|---|---|---|---|---|---|---|---|
| 0x00X0 IDR0 | R | ID28 | ID27 | ID26 | ID25 | ID24 | ID23 | ID22 | ID21 |
| | W | | | | | | | | |
| 0x00X1 IDR1 | R | ID20 | ID19 | ID18 | SRR (=1) | IDE (=1) | ID17 | ID16 | ID15 |
| | W | | | | | | | | |
| 0x00X2 IDR2 | R | ID14 | ID13 | ID12 | ID11 | ID10 | ID9 | ID8 | ID7 |
| | W | | | | | | | | |
| 0x00X3 IDR3 | R | ID6 | ID5 | ID4 | ID3 | ID2 | ID1 | ID0 | RTR |
| | W | | | | | | | | |
| 0x00X4 DSR0 | R | DB7 | DB6 | DB5 | DB4 | DB3 | DB2 | DB1 | DB0 |
| | W | | | | | | | | |
| 0x00X5 DSR1 | R | DB7 | DB6 | DB5 | DB4 | DB3 | DB2 | DB1 | DB0 |
| | W | | | | | | | | |
| 0x00X6 DSR2 | R | DB7 | DB6 | DB5 | DB4 | DB3 | DB2 | DB1 | DB0 |
| | W | | | | | | | | |
| 0x00X7 DSR3 | R | DB7 | DB6 | DB5 | DB4 | DB3 | DB2 | DB1 | DB0 |
| | W | | | | | | | | |
| 0x00X8 DSR4 | R | DB7 | DB6 | DB5 | DB4 | DB3 | DB2 | DB1 | DB0 |
| | W | | | | | | | | |
| 0x00X9 DSR5 | R | DB7 | DB6 | DB5 | DB4 | DB3 | DB2 | DB1 | DB0 |
| | W | | | | | | | | |
| 0x00XA DSR6 | R | DB7 | DB6 | DB5 | DB4 | DB3 | DB2 | DB1 | DB0 |
| | W | | | | | | | | |
| 0x00XB DSR7 | R | DB7 | DB6 | DB5 | DB4 | DB3 | DB2 | DB1 | DB0 |
| | W | | | | | | | | |
| 0x00XC DLR | R | | | | | DLC3 | DLC2 | DLC1 | DLC0 |
| | W | | | | | | | | |

**Figure 16-24. Receive/Transmit Message Buffer — Extended Identifier Mapping (continued)**

| Register Name | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit0 |
|---|---|---|---|---|---|---|---|---|

= Unused, always read 'x'

Read:

- For transmit buffers, anytime when TXEx flag is set (see Section 16.3.2.7, "MSCAN Transmitter Flag Register (CANTFLG)") and the corresponding transmit buffer is selected in CANTBSEL (see Section 16.3.2.11, "MSCAN Transmit Buffer Selection Register (CANTBSEL)").
- For receive buffers, only when RXF flag is set (see Section 16.3.2.5, "MSCAN Receiver Flag Register (CANRFLG)").

Write:

- For transmit buffers, anytime when TXEx flag is set (see Section 16.3.2.7, "MSCAN Transmitter Flag Register (CANTFLG)") and the corresponding transmit buffer is selected in CANTBSEL (see Section 16.3.2.11, "MSCAN Transmit Buffer Selection Register (CANTBSEL)").
- Unimplemented for receive buffers.

Reset: Undefined because of RAM-based implementation

**Figure 16-25. Receive/Transmit Message Buffer — Standard Identifier Mapping**

| Register Name | | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|---|
| IDR0 0x00X0 | R | ID10 | ID9 | ID8 | ID7 | ID6 | ID5 | ID4 | ID3 |
| | W | | | | | | | | |
| IDR1 0x00X1 | R | ID2 | ID1 | ID0 | RTR | IDE (=0) | | | |
| | W | | | | | | | | |
| IDR2 0x00X2 | R | | | | | | | | |
| | W | | | | | | | | |
| IDR3 0x00X3 | R | | | | | | | | |
| | W | | | | | | | | |

= Unused, always read 'x'

### 16.3.3.1 Identifier Registers (IDR0–IDR3)

The identifier registers for an extended format identifier consist of a total of 32 bits: ID[28:0], SRR, IDE, and RTR. The identifier registers for a standard format identifier consist of a total of 13 bits: ID[10:0], RTR, and IDE.

#### 16.3.3.1.1 IDR0–IDR3 for Extended Identifier Mapping

Module Base + 0x00X0

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | ID28 | ID27 | ID26 | ID25 | ID24 | ID23 | ID22 | ID21 |
| Reset: | x | x | x | x | x | x | x | x |

**Figure 16-26. Identifier Register 0 (IDR0) — Extended Identifier Mapping**

**Table 16-27. IDR0 Register Field Descriptions — Extended**

| Field | Description |
|---|---|
| 7-0<br>ID[28:21] | **Extended Format Identifier —** The identifiers consist of 29 bits (ID[28:0]) for the extended format. ID28 is the most significant bit and is transmitted first on the CAN bus during the arbitration procedure. The priority of an identifier is defined to be highest for the smallest binary number. |

Module Base + 0x00X1

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | ID20 | ID19 | ID18 | SRR (=1) | IDE (=1) | ID17 | ID16 | ID15 |
| Reset: | x | x | x | x | x | x | x | x |

**Figure 16-27. Identifier Register 1 (IDR1) — Extended Identifier Mapping**

**Table 16-28. IDR1 Register Field Descriptions — Extended**

| Field | Description |
|---|---|
| 7-5<br>ID[20:18] | **Extended Format Identifier —** The identifiers consist of 29 bits (ID[28:0]) for the extended format. ID28 is the most significant bit and is transmitted first on the CAN bus during the arbitration procedure. The priority of an identifier is defined to be highest for the smallest binary number. |
| 4<br>SRR | **Substitute Remote Request** — This fixed recessive bit is used only in extended format. It must be set to 1 by the user for transmission buffers and is stored as received on the CAN bus for receive buffers. |
| 3<br>IDE | **ID Extended** — This flag indicates whether the extended or standard identifier format is applied in this buffer. In the case of a receive buffer, the flag is set as received and indicates to the CPU how to process the buffer identifier registers. In the case of a transmit buffer, the flag indicates to the MSCAN what type of identifier to send.<br>0 Standard format (11 bit)<br>1 Extended format (29 bit) |
| 2-0<br>ID[17:15] | **Extended Format Identifier —** The identifiers consist of 29 bits (ID[28:0]) for the extended format. ID28 is the most significant bit and is transmitted first on the CAN bus during the arbitration procedure. The priority of an identifier is defined to be highest for the smallest binary number. |

Module Base + 0x00X2

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | ID14 | ID13 | ID12 | ID11 | ID10 | ID9 | ID8 | ID7 |
| Reset: | x | x | x | x | x | x | x | x |

**Figure 16-28. Identifier Register 2 (IDR2) — Extended Identifier Mapping**

**Table 16-29. IDR2 Register Field Descriptions — Extended**

| Field | Description |
|---|---|
| 7-0<br>ID[14:7] | **Extended Format Identifier —** The identifiers consist of 29 bits (ID[28:0]) for the extended format. ID28 is the most significant bit and is transmitted first on the CAN bus during the arbitration procedure. The priority of an identifier is defined to be highest for the smallest binary number. |

Module Base + 0x00X3

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | ID6 | ID5 | ID4 | ID3 | ID2 | ID1 | ID0 | RTR |
| Reset: | x | x | x | x | x | x | x | x |

**Figure 16-29. Identifier Register 3 (IDR3) — Extended Identifier Mapping**

**Table 16-30. IDR3 Register Field Descriptions — Extended**

| Field | Description |
|---|---|
| 7-1<br>ID[6:0] | **Extended Format Identifier —** The identifiers consist of 29 bits (ID[28:0]) for the extended format. ID28 is the most significant bit and is transmitted first on the CAN bus during the arbitration procedure. The priority of an identifier is defined to be highest for the smallest binary number. |
| 0<br>RTR | **Remote Transmission Request** — This flag reflects the status of the remote transmission request bit in the CAN frame. In the case of a receive buffer, it indicates the status of the received frame and supports the transmission of an answering frame in software. In the case of a transmit buffer, this flag defines the setting of the RTR bit to be sent.<br>0 Data frame<br>1 Remote frame |

### 16.3.3.1.2 IDR0–IDR3 for Standard Identifier Mapping

Module Base + 0x00X0

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | ID10 | ID9 | ID8 | ID7 | ID6 | ID5 | ID4 | ID3 |
| Reset: | x | x | x | x | x | x | x | x |

**Figure 16-30. Identifier Register 0 — Standard Mapping**

**Table 16-31. IDR0 Register Field Descriptions — Standard**

| Field | Description |
|---|---|
| 7-0<br>ID[10:3] | **Standard Format Identifier —** The identifiers consist of 11 bits (ID[10:0]) for the standard format. ID10 is the most significant bit and is transmitted first on the CAN bus during the arbitration procedure. The priority of an identifier is defined to be highest for the smallest binary number. See also ID bits in Table 16-32. |

Module Base + 0x00X1

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | ID2 | ID1 | ID0 | RTR | IDE (=0) | | | |
| Reset: | x | x | x | x | x | x | x | x |

= Unused; always read 'x'

**Figure 16-31. Identifier Register 1 — Standard Mapping**

**Table 16-32. IDR1 Register Field Descriptions**

| Field | Description |
|---|---|
| 7-5<br>ID[2:0] | **Standard Format Identifier —** The identifiers consist of 11 bits (ID[10:0]) for the standard format. ID10 is the most significant bit and is transmitted first on the CAN bus during the arbitration procedure. The priority of an identifier is defined to be highest for the smallest binary number. See also ID bits in Table 16-31. |
| 4<br>RTR | **Remote Transmission Request** — This flag reflects the status of the Remote Transmission Request bit in the CAN frame. In the case of a receive buffer, it indicates the status of the received frame and supports the transmission of an answering frame in software. In the case of a transmit buffer, this flag defines the setting of the RTR bit to be sent.<br>0 Data frame<br>1 Remote frame |
| 3<br>IDE | **ID Extended** — This flag indicates whether the extended or standard identifier format is applied in this buffer. In the case of a receive buffer, the flag is set as received and indicates to the CPU how to process the buffer identifier registers. In the case of a transmit buffer, the flag indicates to the MSCAN what type of identifier to send.<br>0 Standard format (11 bit)<br>1 Extended format (29 bit) |

Module Base + 0x00X2

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | | | | | | | | |
| Reset: | x | x | x | x | x | x | x | x |

= Unused; always read 'x'

**Figure 16-32. Identifier Register 2 — Standard Mapping**

Module Base + 0x00X3

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | | | | | | | | |
| Reset: | x | x | x | x | x | x | x | x |

= Unused; always read 'x'

**Figure 16-33. Identifier Register 3 — Standard Mapping**

### 16.3.3.2 Data Segment Registers (DSR0-7)

The eight data segment registers, each with bits DB[7:0], contain the data to be transmitted or received. The number of bytes to be transmitted or received is determined by the data length code in the corresponding DLR register.

Module Base + 0x00X4 to Module Base + 0x00XB

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | DB7 | DB6 | DB5 | DB4 | DB3 | DB2 | DB1 | DB0 |
| Reset: | x | x | x | x | x | x | x | x |

**Figure 16-34. Data Segment Registers (DSR0–DSR7) — Extended Identifier Mapping**

**Table 16-33. DSR0–DSR7 Register Field Descriptions**

| Field | Description |
|---|---|
| 7-0<br>DB[7:0] | **Data bits 7-0** |

### 16.3.3.3 Data Length Register (DLR)

This register keeps the data length field of the CAN frame.

Module Base + 0x00XC

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | | | | | DLC3 | DLC2 | DLC1 | DLC0 |
| Reset: | x | x | x | x | x | x | x | x |

= Unused; always read "x"

**Figure 16-35. Data Length Register (DLR) — Extended Identifier Mapping**

**Table 16-34. DLR Register Field Descriptions**

| Field | Description |
|---|---|
| 3-0<br>DLC[3:0] | **Data Length Code Bits** — The data length code contains the number of bytes (data byte count) of the respective message. During the transmission of a remote frame, the data length code is transmitted as programmed while the number of transmitted data bytes is always 0. The data byte count ranges from 0 to 8 for a data frame. Table 16-35 shows the effect of setting the DLC bits. |

**Table 16-35. Data Length Codes**

| Data Length Code | | | | Data Byte Count |
|---|---|---|---|---|
| DLC3 | DLC2 | DLC1 | DLC0 | |
| 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 1 | 1 |
| 0 | 0 | 1 | 0 | 2 |
| 0 | 0 | 1 | 1 | 3 |
| 0 | 1 | 0 | 0 | 4 |
| 0 | 1 | 0 | 1 | 5 |
| 0 | 1 | 1 | 0 | 6 |
| 0 | 1 | 1 | 1 | 7 |
| 1 | 0 | 0 | 0 | 8 |

### 16.3.3.4 Transmit Buffer Priority Register (TBPR)

This register defines the local priority of the associated message buffer. The local priority is used for the internal prioritization process of the MSCAN and is defined to be highest for the smallest binary number. The MSCAN implements the following internal prioritization mechanisms:

- All transmission buffers with a cleared TXEx flag participate in the prioritization immediately before the SOF (start of frame) is sent.

- The transmission buffer with the lowest local priority field wins the prioritization.

In cases of more than one buffer having the same lowest priority, the message buffer with the lower index number wins.

Module Base + 0x00XD — Access: User read/write(1)

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | PRIO7 | PRIO6 | PRIO5 | PRIO4 | PRIO3 | PRIO2 | PRIO1 | PRIO0 |
| Reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 16-36. Transmit Buffer Priority Register (TBPR)**

1. Read: Anytime when TXEx flag is set (see Section 16.3.2.7, "MSCAN Transmitter Flag Register (CANTFLG)") and the corresponding transmit buffer is selected in CANTBSEL (see Section 16.3.2.11, "MSCAN Transmit Buffer Selection Register (CANTBSEL)")
   Write: Anytime when TXEx flag is set (see Section 16.3.2.7, "MSCAN Transmitter Flag Register (CANTFLG)") and the corresponding transmit buffer is selected in CANTBSEL (see Section 16.3.2.11, "MSCAN Transmit Buffer Selection Register (CANTBSEL)")

### 16.3.3.5 Time Stamp Register (TSRH–TSRL)

If the TIME bit is enabled, the MSCAN will write a time stamp to the respective registers in the active transmit or receive buffer right after the EOF of a valid message on the CAN bus (see Section 16.3.2.1, "MSCAN Control Register 0 (CANCTL0)"). In case of a transmission, the CPU can only read the time stamp after the respective transmit buffer has been flagged empty.

The timer value, which is used for stamping, is taken from a free running internal CAN bit clock. A timer overrun is not indicated by the MSCAN. The timer is reset (all bits set to 0) during initialization mode. The CPU can only read the time stamp registers.

Module Base + 0x00XE — Access: User read/write(1)

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | TSR15 | TSR14 | TSR13 | TSR12 | TSR11 | TSR10 | TSR9 | TSR8 |
| W | | | | | | | | |
| Reset: | x | x | x | x | x | x | x | x |

**Figure 16-37. Time Stamp Register — High Byte (TSRH)**

1. Read: For transmit buffers: Anytime when TXEx flag is set (see Section 16.3.2.7, "MSCAN Transmitter Flag Register (CANTFLG)") and the corresponding transmit buffer is selected in CANTBSEL (see Section 16.3.2.11, "MSCAN Transmit Buffer Selection Register (CANTBSEL)"). For receive buffers: Anytime when RXF is set.
   Write: Unimplemented

Module Base + 0x00XF

Access: User read/write(1)

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | TSR7 | TSR6 | TSR5 | TSR4 | TSR3 | TSR2 | TSR1 | TSR0 |
| W | | | | | | | | |
| Reset: | x | x | x | x | x | x | x | x |

**Figure 16-38. Time Stamp Register — Low Byte (TSRL)**

1. Read: or transmit buffers: Anytime when TXEx flag is set (see Section 16.3.2.7, "MSCAN Transmitter Flag Register (CANTFLG)") and the corresponding transmit buffer is selected in CANTBSEL (see Section 16.3.2.11, "MSCAN Transmit Buffer Selection Register (CANTBSEL)"). For receive buffers: Anytime when RXF is set.
   Write: Unimplemented

# 16.4 Functional Description

## 16.4.1 General

This section provides a complete functional description of the MSCAN.

## 16.4.2 Message Storage

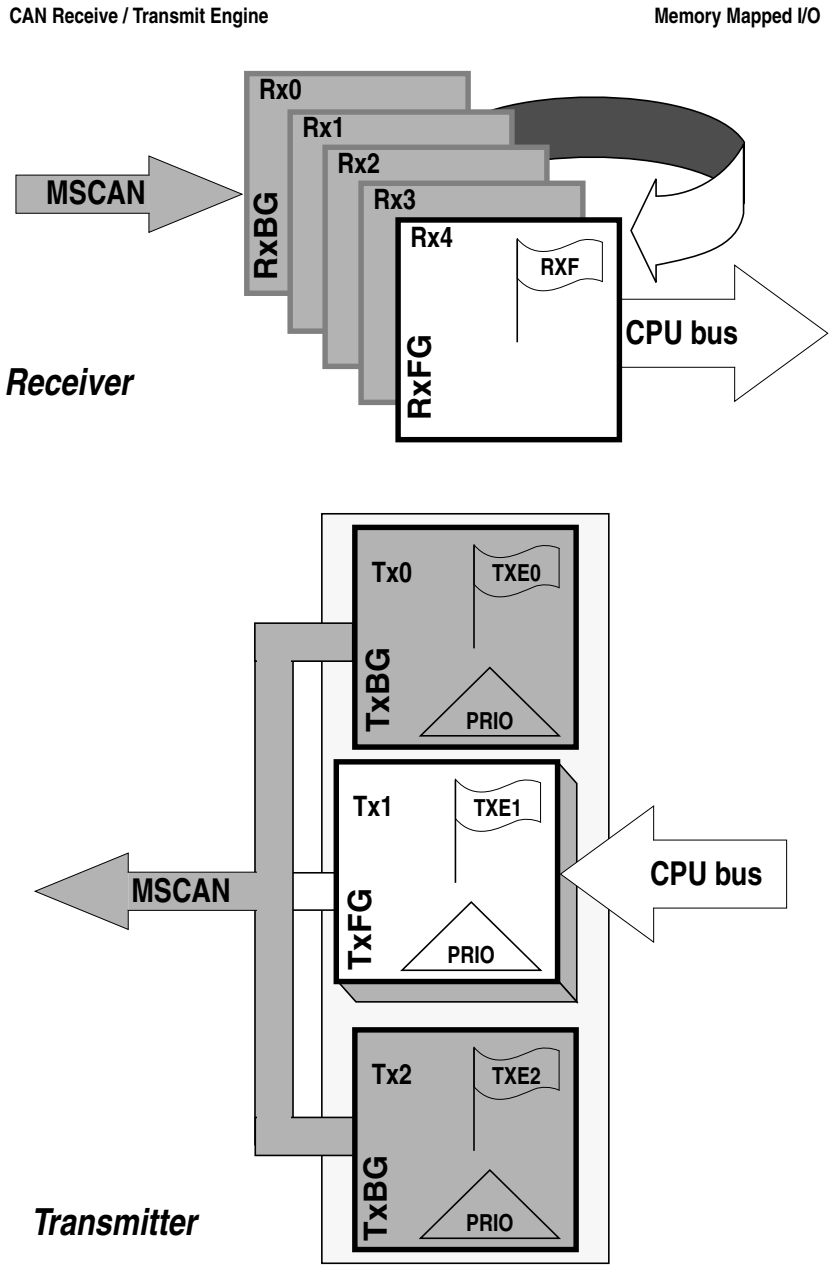



**Figure 16-39. User Model for Message Buffer Organization**

The MSCAN facilitates a sophisticated message storage system which addresses the requirements of a broad range of network applications.

#### 16.4.2.1 Message Transmit Background

Modern application layer software is built upon two fundamental assumptions:

- Any CAN node is able to send out a stream of scheduled messages without releasing the CAN bus between the two messages. Such nodes arbitrate for the CAN bus immediately after sending the previous message and only release the CAN bus in case of lost arbitration.
- The internal message queue within any CAN node is organized such that the highest priority message is sent out first, if more than one message is ready to be sent.

The behavior described in the bullets above cannot be achieved with a single transmit buffer. That buffer must be reloaded immediately after the previous message is sent. This loading process lasts a finite amount of time and must be completed within the inter-frame sequence (IFS) to be able to send an uninterrupted stream of messages. Even if this is feasible for limited CAN bus speeds, it requires that the CPU reacts with short latencies to the transmit interrupt.

A double buffer scheme de-couples the reloading of the transmit buffer from the actual message sending and, therefore, reduces the reactiveness requirements of the CPU. Problems can arise if the sending of a message is finished while the CPU re-loads the second buffer. No buffer would then be ready for transmission, and the CAN bus would be released.

At least three transmit buffers are required to meet the first of the above requirements under all circumstances. The MSCAN has three transmit buffers.

The second requirement calls for some sort of internal prioritization which the MSCAN implements with the "local priority" concept described in Section 16.4.2.2, "Transmit Structures."

#### 16.4.2.2 Transmit Structures

The MSCAN triple transmit buffer scheme optimizes real-time performance by allowing multiple messages to be set up in advance. The three buffers are arranged as shown in Figure 16-39.

All three buffers have a 13-byte data structure similar to the outline of the receive buffers (see Section 16.3.3, "Programmer's Model of Message Storage"). An additional Transmit Buffer Priority Register (TBPR) contains an 8-bit local priority field (PRIO) (see Section 16.3.3.4, "Transmit Buffer Priority Register (TBPR)"). The remaining two bytes are used for time stamping of a message, if required (see Section 16.3.3.5, "Time Stamp Register (TSRH–TSRL)").

To transmit a message, the CPU must identify an available transmit buffer, which is indicated by a set transmitter buffer empty (TXEx) flag (see Section 16.3.2.7, "MSCAN Transmitter Flag Register (CANTFLG)"). If a transmit buffer is available, the CPU must set a pointer to this buffer by writing to the CANTBSEL register (see Section 16.3.2.11, "MSCAN Transmit Buffer Selection Register (CANTBSEL)"). This makes the respective buffer accessible within the CANTXFG address space (see Section 16.3.3, "Programmer's Model of Message Storage"). The algorithmic feature associated with the CANTBSEL register simplifies the transmit buffer selection. In addition, this scheme makes the handler

software simpler because only one address area is applicable for the transmit process, and the required address space is minimized.

The CPU then stores the identifier, the control bits, and the data content into one of the transmit buffers. Finally, the buffer is flagged as ready for transmission by clearing the associated TXE flag.

The MSCAN then schedules the message for transmission and signals the successful transmission of the buffer by setting the associated TXE flag. A transmit interrupt (see Section 16.4.7.2, "Transmit Interrupt") is generated[1] when TXEx is set and can be used to drive the application software to re-load the buffer.

If more than one buffer is scheduled for transmission when the CAN bus becomes available for arbitration, the MSCAN uses the local priority setting of the three buffers to determine the prioritization. For this purpose, every transmit buffer has an 8-bit local priority field (PRIO). The application software programs this field when the message is set up. The local priority reflects the priority of this particular message relative to the set of messages being transmitted from this node. The lowest binary value of the PRIO field is defined to be the highest priority. The internal scheduling process takes place whenever the MSCAN arbitrates for the CAN bus. This is also the case after the occurrence of a transmission error.

When a high priority message is scheduled by the application software, it may become necessary to abort a lower priority message in one of the three transmit buffers. Because messages that are already in transmission cannot be aborted, the user must request the abort by setting the corresponding abort request bit (ABTRQ) (see Section 16.3.2.9, "MSCAN Transmitter Message Abort Request Register (CANTARQ)".) The MSCAN then grants the request, if possible, by:

1. Setting the corresponding abort acknowledge flag (ABTAK) in the CANTAAK register.
2. Setting the associated TXE flag to release the buffer.
3. Generating a transmit interrupt. The transmit interrupt handler software can determine from the setting of the ABTAK flag whether the message was aborted (ABTAK = 1) or sent (ABTAK = 0).

### 16.4.2.3 Receive Structures

The received messages are stored in a five stage input FIFO. The five message buffers are alternately mapped into a single memory area (see Figure 16-39). The background receive buffer (RxBG) is exclusively associated with the MSCAN, but the foreground receive buffer (RxFG) is addressable by the CPU (see Figure 16-39). This scheme simplifies the handler software because only one address area is applicable for the receive process.

All receive buffers have a size of 15 bytes to store the CAN control bits, the identifier (standard or extended), the data contents, and a time stamp, if enabled (see Section 16.3.3, "Programmer's Model of Message Storage").

The receiver full flag (RXF) (see Section 16.3.2.5, "MSCAN Receiver Flag Register (CANRFLG)") signals the status of the foreground receive buffer. When the buffer contains a correctly received message with a matching identifier, this flag is set.

On reception, each message is checked to see whether it passes the filter (see Section 16.4.3, "Identifier Acceptance Filter") and simultaneously is written into the active RxBG. After successful reception of a valid message, the MSCAN shifts the content of RxBG into the receiver FIFO, sets the RXF flag, and

1. The transmit interrupt occurs only if not masked. A polling scheme can be applied on TXEx also.

generates a receive interrupt[1] (see Section 16.4.7.3, "Receive Interrupt") to the CPU. The user's receive handler must read the received message from the RxFG and then reset the RXF flag to acknowledge the interrupt and to release the foreground buffer. A new message, which can follow immediately after the IFS field of the CAN frame, is received into the next available RxBG. If the MSCAN receives an invalid message in its RxBG (wrong identifier, transmission errors, etc.) the actual contents of the buffer will be over-written by the next message. The buffer will then not be shifted into the FIFO.

When the MSCAN module is transmitting, the MSCAN receives its own transmitted messages into the background receive buffer, RxBG, but does not shift it into the receiver FIFO, generate a receive interrupt, or acknowledge its own messages on the CAN bus. The exception to this rule is in loopback mode (see Section 16.3.2.2, "MSCAN Control Register 1 (CANCTL1)") where the MSCAN treats its own messages exactly like all other incoming messages. The MSCAN receives its own transmitted messages in the event that it loses arbitration. If arbitration is lost, the MSCAN must be prepared to become a receiver.

An overrun condition occurs when all receive message buffers in the FIFO are filled with correctly received messages with accepted identifiers and another message is correctly received from the CAN bus with an accepted identifier. The latter message is discarded and an error interrupt with overrun indication is generated if enabled (see Section 16.4.7.5, "Error Interrupt"). The MSCAN remains able to transmit messages while the receiver FIFO is being filled, but all incoming messages are discarded. As soon as a receive buffer in the FIFO is available again, new valid messages will be accepted.

### 16.4.3 Identifier Acceptance Filter

The MSCAN identifier acceptance registers (see Section 16.3.2.12, "MSCAN Identifier Acceptance Control Register (CANIDAC)") define the acceptable patterns of the standard or extended identifier (ID[10:0] or ID[28:0]). Any of these bits can be marked 'don't care' in the MSCAN identifier mask registers (see Section 16.3.2.18, "MSCAN Identifier Mask Registers (CANIDMR0–CANIDMR7)").

A filter hit is indicated to the application software by a set receive buffer full flag (RXF = 1) and three bits in the CANIDAC register (see Section 16.3.2.12, "MSCAN Identifier Acceptance Control Register (CANIDAC)"). These identifier hit flags (IDHIT[2:0]) clearly identify the filter section that caused the acceptance. They simplify the application software's task to identify the cause of the receiver interrupt. If more than one hit occurs (two or more filters match), the lower hit has priority.

A very flexible programmable generic identifier acceptance filter has been introduced to reduce the CPU interrupt loading. The filter is programmable to operate in four different modes:

- Two identifier acceptance filters, each to be applied to:
  - The full 29 bits of the extended identifier and to the following bits of the CAN 2.0B frame:
    - Remote transmission request (RTR)
    - Identifier extension (IDE)
    - Substitute remote request (SRR)
  - The 11 bits of the standard identifier plus the RTR and IDE bits of the CAN 2.0A/B messages. This mode implements two filters for a full length CAN 2.0B compliant extended identifier. Although this mode can be used for standard identifiers, it is recommended to use the four or eight identifier acceptance filters.

1. The receive interrupt occurs only if not masked. A polling scheme can be applied on RXF also.

Figure 16-40 shows how the first 32-bit filter bank (CANIDAR0–CANIDAR3, CANIDMR0–CANIDMR3) produces a filter 0 hit. Similarly, the second filter bank (CANIDAR4–CANIDAR7, CANIDMR4–CANIDMR7) produces a filter 1 hit.

- Four identifier acceptance filters, each to be applied to:
  - The 14 most significant bits of the extended identifier plus the SRR and IDE bits of CAN 2.0B messages.
  - The 11 bits of the standard identifier, the RTR and IDE bits of CAN 2.0A/B messages. Figure 16-41 shows how the first 32-bit filter bank (CANIDAR0–CANIDAR3, CANIDMR0–CANIDMR3) produces filter 0 and 1 hits. Similarly, the second filter bank (CANIDAR4–CANIDAR7, CANIDMR4–CANIDMR7) produces filter 2 and 3 hits.
- Eight identifier acceptance filters, each to be applied to the first 8 bits of the identifier. This mode implements eight independent filters for the first 8 bits of a CAN 2.0A/B compliant standard identifier or a CAN 2.0B compliant extended identifier.
  Figure 16-42 shows how the first 32-bit filter bank (CANIDAR0–CANIDAR3, CANIDMR0–CANIDMR3) produces filter 0 to 3 hits. Similarly, the second filter bank (CANIDAR4–CANIDAR7, CANIDMR4–CANIDMR7) produces filter 4 to 7 hits.
- Closed filter. No CAN message is copied into the foreground buffer RxFG, and the RXF flag is never set.

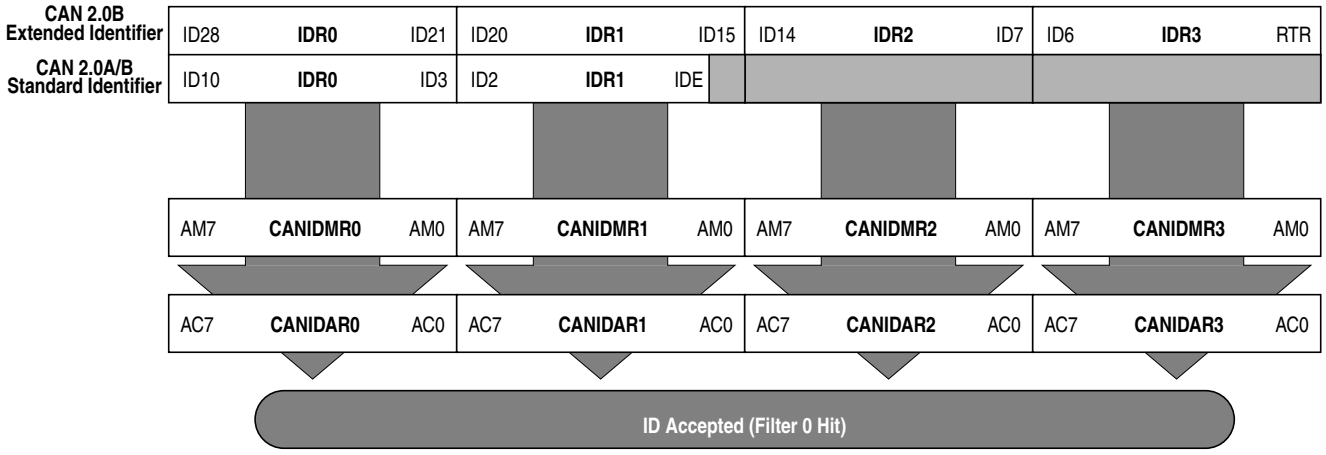



**Figure 16-40. 32-bit Maskable Identifier Acceptance Filter**

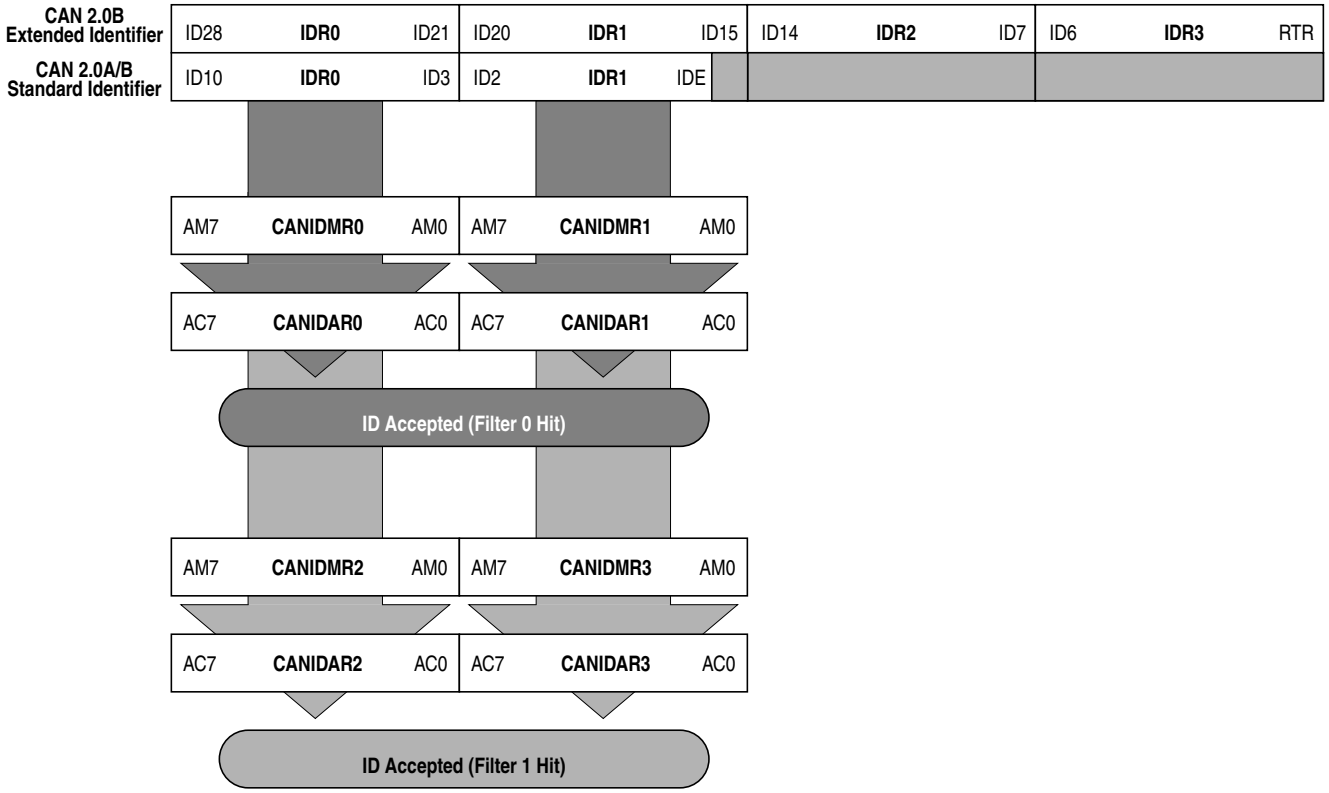



**Figure 16-41. 16-bit Maskable Identifier Acceptance Filters**

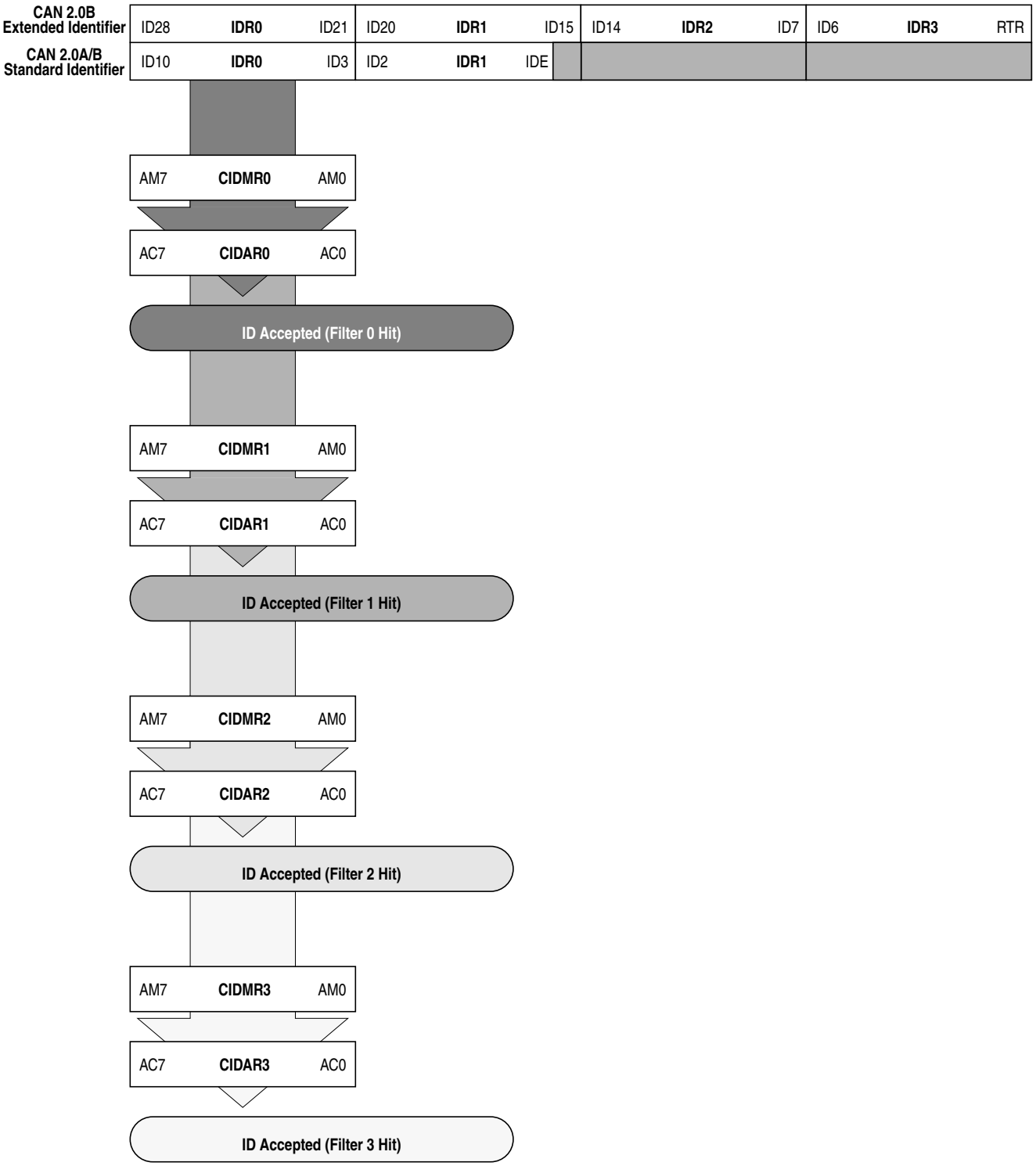



**Figure 16-42. 8-bit Maskable Identifier Acceptance Filters**

### 16.4.3.1 Protocol Violation Protection

The MSCAN protects the user from accidentally violating the CAN protocol through programming errors. The protection logic implements the following features:

- The receive and transmit error counters cannot be written or otherwise manipulated.
- All registers which control the configuration of the MSCAN cannot be modified while the MSCAN is on-line. The MSCAN has to be in Initialization Mode. The corresponding INITRQ/INITAK handshake bits in the CANCTL0/CANCTL1 registers (see Section 16.3.2.1, "MSCAN Control Register 0 (CANCTL0)") serve as a lock to protect the following registers:
  - MSCAN control 1 register (CANCTL1)
  - MSCAN bus timing registers 0 and 1 (CANBTR0, CANBTR1)
  - MSCAN identifier acceptance control register (CANIDAC)
  - MSCAN identifier acceptance registers (CANIDAR0–CANIDAR7)
  - MSCAN identifier mask registers (CANIDMR0–CANIDMR7)
- The TXCAN is immediately forced to a recessive state when the MSCAN goes into the power down mode or initialization mode (see Section 16.4.5.6, "MSCAN Power Down Mode," and Section 16.4.4.5, "MSCAN Initialization Mode").
- The MSCAN enable bit (CANE) is writable only once in normal system operation modes, which provides further protection against inadvertently disabling the MSCAN.

### 16.4.3.2 Clock System

Figure 16-43 shows the structure of the MSCAN clock generation circuitry.

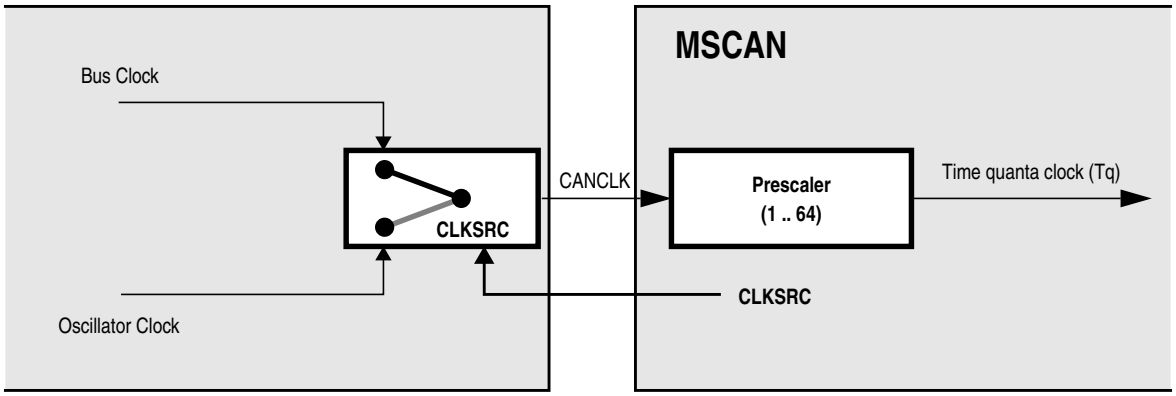



**Figure 16-43. MSCAN Clocking Scheme**

The clock source bit (CLKSRC) in the CANCTL1 register (16.3.2.2/16-613) defines whether the internal CANCLK is connected to the output of a crystal oscillator (oscillator clock) or to the bus clock.

The clock source has to be chosen such that the tight oscillator tolerance requirements (up to 0.4%) of the CAN protocol are met. Additionally, for high CAN bus rates (1 Mbps), a 45% to 55% duty cycle of the clock is required.

If the bus clock is generated from a PLL, it is recommended to select the oscillator clock rather than the bus clock due to jitter considerations, especially at the faster CAN bus rates.

For microcontrollers without a clock and reset generator (CRG), CANCLK is driven from the crystal oscillator (oscillator clock).

A programmable prescaler generates the time quanta (Tq) clock from CANCLK. A time quantum is the atomic unit of time handled by the MSCAN.

***Eqn. 16-2***

$$Tq = \frac{f_{CANCLK}}{(\text{Prescaler value})}$$

A bit time is subdivided into three segments as described in the Bosch CAN 2.0A/B specification. (see Figure 16-44):

- SYNC_SEG: This segment has a fixed length of one time quantum. Signal edges are expected to happen within this section.
- Time Segment 1: This segment includes the PROP_SEG and the PHASE_SEG1 of the CAN standard. It can be programmed by setting the parameter TSEG1 to consist of 4 to 16 time quanta.
- Time Segment 2: This segment represents the PHASE_SEG2 of the CAN standard. It can be programmed by setting the TSEG2 parameter to be 2 to 8 time quanta long.

***Eqn. 16-3***

$$\text{Bit Rate} = \frac{f_{Tq}}{(\text{number of Time Quanta})}$$

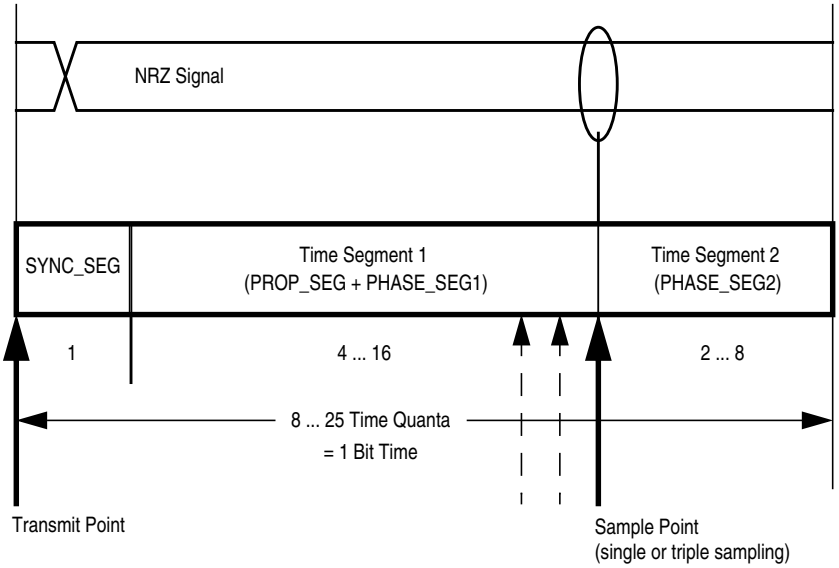



**Figure 16-44. Segments within the Bit Time**

**Table 16-36. Time Segment Syntax**

| Syntax | Description |
|---|---|
| SYNC_SEG | System expects transitions to occur on the CAN bus during this period. |
| Transmit Point | A node in transmit mode transfers a new value to the CAN bus at this point. |
| Sample Point | A node in receive mode samples the CAN bus at this point. If the three samples per bit option is selected, then this point marks the position of the third sample. |

The synchronization jump width (see the Bosch CAN 2.0A/B specification for details) can be programmed in a range of 1 to 4 time quanta by setting the SJW parameter.

The SYNC_SEG, TSEG1, TSEG2, and SJW parameters are set by programming the MSCAN bus timing registers (CANBTR0, CANBTR1) (see Section 16.3.2.3, "MSCAN Bus Timing Register 0 (CANBTR0)" and Section 16.3.2.4, "MSCAN Bus Timing Register 1 (CANBTR1)").

Table 16-37 gives an overview of the Bosch CAN 2.0A/B specification compliant segment settings and the related parameter values.

**NOTE**

It is the user's responsibility to ensure the bit time settings are in compliance with the CAN standard.

**Table 16-37. Bosch CAN 2.0A/B Compliant Bit Time Segment Settings**

| Time Segment 1 | TSEG1 | Time Segment 2 | TSEG2 | Synchronization Jump Width | SJW |
|---|---|---|---|---|---|
| 5 .. 10 | 4 .. 9 | 2 | 1 | 1 .. 2 | 0 .. 1 |
| 4 .. 11 | 3 .. 10 | 3 | 2 | 1 .. 3 | 0 .. 2 |
| 5 .. 12 | 4 .. 11 | 4 | 3 | 1 .. 4 | 0 .. 3 |
| 6 .. 13 | 5 .. 12 | 5 | 4 | 1 .. 4 | 0 .. 3 |
| 7 .. 14 | 6 .. 13 | 6 | 5 | 1 .. 4 | 0 .. 3 |
| 8 .. 15 | 7 .. 14 | 7 | 6 | 1 .. 4 | 0 .. 3 |
| 9 .. 16 | 8 .. 15 | 8 | 7 | 1 .. 4 | 0 .. 3 |

## 16.4.4 Modes of Operation

### 16.4.4.1 Normal System Operating Modes

The MSCAN module behaves as described within this specification in all normal system operating modes. Write restrictions exist for some registers.

#### 16.4.4.2 Special System Operating Modes

The MSCAN module behaves as described within this specification in all special system operating modes. Write restrictions which exist on specific registers in normal modes are lifted for test purposes in special modes.

#### 16.4.4.3 Emulation Modes

In all emulation modes, the MSCAN module behaves just like in normal system operating modes as described within this specification.

#### 16.4.4.4 Listen-Only Mode

In an optional CAN bus monitoring mode (listen-only), the CAN node is able to receive valid data frames and valid remote frames, but it sends only "recessive" bits on the CAN bus. In addition, it cannot start a transmission.

If the MAC sub-layer is required to send a "dominant" bit (ACK bit, overload flag, or active error flag), the bit is rerouted internally so that the MAC sub-layer monitors this "dominant" bit, although the CAN bus may remain in recessive state externally.

#### 16.4.4.5 MSCAN Initialization Mode

The MSCAN enters initialization mode when it is enabled (CANE=1).

When entering initialization mode during operation, any on-going transmission or reception is immediately aborted and synchronization to the CAN bus is lost, potentially causing CAN protocol violations. To protect the CAN bus system from fatal consequences of violations, the MSCAN immediately drives TXCAN into a recessive state.

**NOTE**

The user is responsible for ensuring that the MSCAN is not active when initialization mode is entered. The recommended procedure is to bring the MSCAN into sleep mode (SLPRQ = 1 and SLPAK = 1) before setting the INITRQ bit in the CANCTL0 register. Otherwise, the abort of an on-going message can cause an error condition and can impact other CAN bus devices.

In initialization mode, the MSCAN is stopped. However, interface registers remain accessible. This mode is used to reset the CANCTL0, CANRFLG, CANRIER, CANTFLG, CANTIER, CANTARQ, CANTAAK, and CANTBSEL registers to their default values. In addition, the MSCAN enables the configuration of the CANBTR0, CANBTR1 bit timing registers; CANIDAC; and the CANIDAR, CANIDMR message filters. See Section 16.3.2.1, "MSCAN Control Register 0 (CANCTL0)," for a detailed description of the initialization mode.

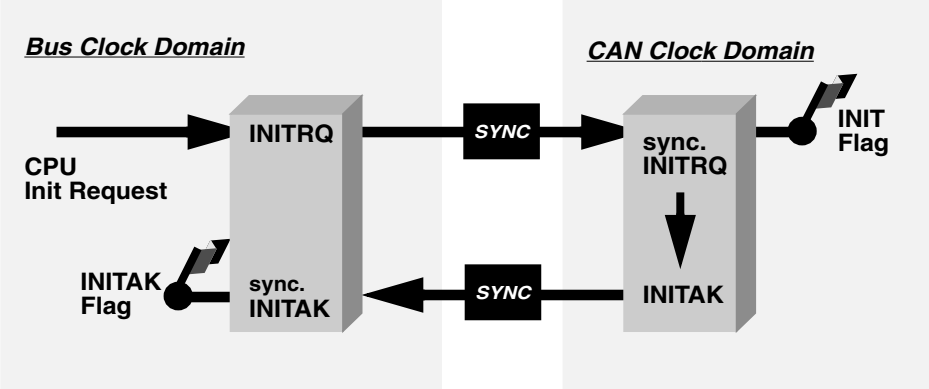



**Figure 16-45. Initialization Request/Acknowledge Cycle**

Due to independent clock domains within the MSCAN, INITRQ must be synchronized to all domains by using a special handshake mechanism. This handshake causes additional synchronization delay (see Figure 16-45).

If there is no message transfer ongoing on the CAN bus, the minimum delay will be two additional bus clocks and three additional CAN clocks. When all parts of the MSCAN are in initialization mode, the INITAK flag is set. The application software must use INITAK as a handshake indication for the request (INITRQ) to go into initialization mode.

**NOTE**

The CPU cannot clear INITRQ before initialization mode (INITRQ = 1 and INITAK = 1) is active.

## 16.4.5 Low-Power Options

If the MSCAN is disabled (CANE = 0), the MSCAN clocks are stopped for power saving.

If the MSCAN is enabled (CANE = 1), the MSCAN has two additional modes with reduced power consumption, compared to normal mode: sleep and power down mode. In sleep mode, power consumption is reduced by stopping all clocks except those to access the registers from the CPU side. In power down mode, all clocks are stopped and no power is consumed.

Table 16-38 summarizes the combinations of MSCAN and CPU modes. A particular combination of modes is entered by the given settings on the CSWAI and SLPRQ/SLPAK bits.

**Table 16-38. CPU vs. MSCAN Operating Modes**

| CPU Mode | MSCAN Mode | | | |
|---|---|---|---|---|
| | Normal | Reduced Power Consumption | | |
| | | Sleep | Power Down | Disabled (CANE=0) |
| **RUN** | CSWAI = X[1]<br>SLPRQ = 0<br>SLPAK = 0 | CSWAI = X<br>SLPRQ = 1<br>SLPAK = 1 | | CSWAI = X<br>SLPRQ = X<br>SLPAK = X |
| **WAIT** | CSWAI = 0<br>SLPRQ = 0<br>SLPAK = 0 | CSWAI = 0<br>SLPRQ = 1<br>SLPAK = 1 | CSWAI = 1<br>SLPRQ = X<br>SLPAK = X | CSWAI = X<br>SLPRQ = X<br>SLPAK = X |
| **STOP** | | | CSWAI = X<br>SLPRQ = X<br>SLPAK = X | CSWAI = X<br>SLPRQ = X<br>SLPAK = X |

1. 'X' means don't care.

### 16.4.5.1 Operation in Run Mode

As shown in Table 16-38, only MSCAN sleep mode is available as low power option when the CPU is in run mode.

### 16.4.5.2 Operation in Wait Mode

The WAI instruction puts the MCU in a low power consumption stand-by mode. If the CSWAI bit is set, additional power can be saved in power down mode because the CPU clocks are stopped. After leaving this power down mode, the MSCAN restarts and enters normal mode again.

While the CPU is in wait mode, the MSCAN can be operated in normal mode and generate interrupts (registers can be accessed via background debug mode).

### 16.4.5.3 Operation in Stop Mode

The STOP instruction puts the MCU in a low power consumption stand-by mode. In stop mode, the MSCAN is set in power down mode regardless of the value of the SLPRQ/SLPAK and CSWAI bits (Table 16-38).

### 16.4.5.4 MSCAN Normal Mode

This is a non-power-saving mode. Enabling the MSCAN puts the module from disabled mode into normal mode. In this mode the module can either be in initialization mode or out of initialization mode. See Section 16.4.4.5, "MSCAN Initialization Mode".

### 16.4.5.5 MSCAN Sleep Mode

The CPU can request the MSCAN to enter this low power mode by asserting the SLPRQ bit in the CANCTL0 register. The time when the MSCAN enters sleep mode depends on a fixed synchronization delay and its current activity:

- If there are one or more message buffers scheduled for transmission (TXEx = 0), the MSCAN will continue to transmit until all transmit message buffers are empty (TXEx = 1, transmitted successfully or aborted) and then goes into sleep mode.
- If the MSCAN is receiving, it continues to receive and goes into sleep mode as soon as the CAN bus next becomes idle.
- If the MSCAN is neither transmitting nor receiving, it immediately goes into sleep mode.

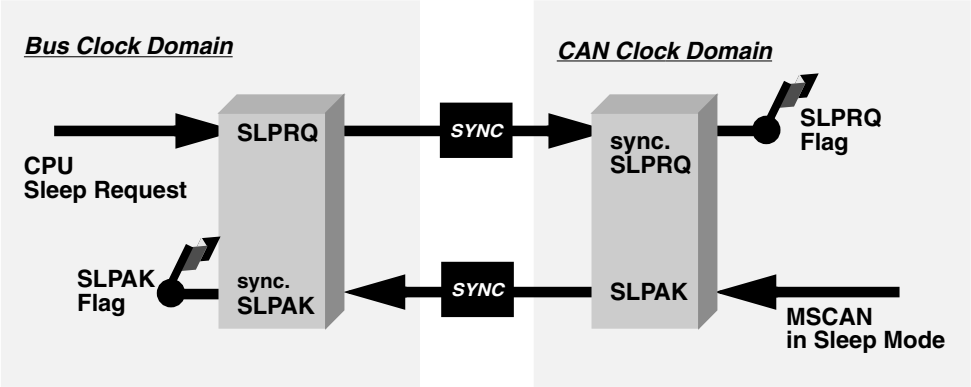



**Figure 16-46. Sleep Request / Acknowledge Cycle**

**NOTE**

The application software must avoid setting up a transmission (by clearing one or more TXEx flag(s)) and immediately request sleep mode (by setting SLPRQ). Whether the MSCAN starts transmitting or goes into sleep mode directly depends on the exact sequence of operations.

If sleep mode is active, the SLPRQ and SLPAK bits are set (Figure 16-46). The application software must use SLPAK as a handshake indication for the request (SLPRQ) to go into sleep mode.

When in sleep mode (SLPRQ = 1 and SLPAK = 1), the MSCAN stops its internal clocks. However, clocks that allow register accesses from the CPU side continue to run.

If the MSCAN is in bus-off state, it stops counting the 128 occurrences of 11 consecutive recessive bits due to the stopped clocks. TXCAN remains in a recessive state. If RXF = 1, the message can be read and RXF can be cleared. Shifting a new message into the foreground buffer of the receiver FIFO (RxFG) does not take place while in sleep mode.

It is possible to access the transmit buffers and to clear the associated TXE flags. No message abort takes place while in sleep mode.

If the WUPE bit in CANCTL0 is not asserted, the MSCAN will mask any activity it detects on CAN. RXCAN is therefore held internally in a recessive state. This locks the MSCAN in sleep mode. WUPE must be set before entering sleep mode to take effect.

The MSCAN is able to leave sleep mode (wake up) only when:

- CAN bus activity occurs and WUPE = 1

  or
- the CPU clears the SLPRQ bit

**NOTE**

The CPU cannot clear the SLPRQ bit before sleep mode (SLPRQ = 1 and SLPAK = 1) is active.

After wake-up, the MSCAN waits for 11 consecutive recessive bits to synchronize to the CAN bus. As a consequence, if the MSCAN is woken-up by a CAN frame, this frame is not received.

The receive message buffers (RxFG and RxBG) contain messages if they were received before sleep mode was entered. All pending actions will be executed upon wake-up; copying of RxBG into RxFG, message aborts and message transmissions. If the MSCAN remains in bus-off state after sleep mode was exited, it continues counting the 128 occurrences of 11 consecutive recessive bits.

#### 16.4.5.6 MSCAN Power Down Mode

The MSCAN is in power down mode (Table 16-38) when

- CPU is in stop mode

  or
- CPU is in wait mode and the CSWAI bit is set

When entering the power down mode, the MSCAN immediately stops all ongoing transmissions and receptions, potentially causing CAN protocol violations. To protect the CAN bus system from fatal consequences of violations to the above rule, the MSCAN immediately drives TXCAN into a recessive state.

**NOTE**

The user is responsible for ensuring that the MSCAN is not active when power down mode is entered. The recommended procedure is to bring the MSCAN into Sleep mode before the STOP or WAI instruction (if CSWAI is set) is executed. Otherwise, the abort of an ongoing message can cause an error condition and impact other CAN bus devices.

In power down mode, all clocks are stopped and no registers can be accessed. If the MSCAN was not in sleep mode before power down mode became active, the module performs an internal recovery cycle after powering up. This causes some fixed delay before the module enters normal mode again.

#### 16.4.5.7 Disabled Mode

The MSCAN is in disabled mode out of reset (CANE=0). All module clocks are stopped for power saving, however the register map can still be accessed as specified.

#### 16.4.5.8 Programmable Wake-Up Function

The MSCAN can be programmed to wake up from sleep or power down mode as soon as CAN bus activity is detected (see control bit WUPE in MSCAN Control Register 0 (CANCTL0). The sensitivity to existing CAN bus action can be modified by applying a low-pass filter function to the RXCAN input line (see control bit WUPM in Section 16.3.2.2, "MSCAN Control Register 1 (CANCTL1)").

This feature can be used to protect the MSCAN from wake-up due to short glitches on the CAN bus lines. Such glitches can result from—for example—electromagnetic interference within noisy environments.

### 16.4.6 Reset Initialization

The reset state of each individual bit is listed in Section 16.3.2, "Register Descriptions," which details all the registers and their bit-fields.

### 16.4.7 Interrupts

This section describes all interrupts originated by the MSCAN. It documents the enable bits and generated flags. Each interrupt is listed and described separately.

#### 16.4.7.1 Description of Interrupt Operation

The MSCAN supports four interrupt vectors (see Table 16-39), any of which can be individually masked (for details see Section 16.3.2.6, "MSCAN Receiver Interrupt Enable Register (CANRIER)" to Section 16.3.2.8, "MSCAN Transmitter Interrupt Enable Register (CANTIER)").

Refer to the device overview section to determine the dedicated interrupt vector addresses.

**Table 16-39. Interrupt Vectors**

| Interrupt Source | CCR Mask | Local Enable |
|---|---|---|
| Wake-Up Interrupt (WUPIF) | I bit | CANRIER (WUPIE) |
| Error Interrupts Interrupt (CSCIF, OVRIF) | I bit | CANRIER (CSCIE, OVRIE) |
| Receive Interrupt (RXF) | I bit | CANRIER (RXFIE) |
| Transmit Interrupts (TXE[2:0]) | I bit | CANTIER (TXEIE[2:0]) |

#### 16.4.7.2 Transmit Interrupt

At least one of the three transmit buffers is empty (not scheduled) and can be loaded to schedule a message for transmission. The TXEx flag of the empty message buffer is set.

#### 16.4.7.3 Receive Interrupt

A message is successfully received and shifted into the foreground buffer (RxFG) of the receiver FIFO. This interrupt is generated immediately after receiving the EOF symbol. The RXF flag is set. If there are multiple messages in the receiver FIFO, the RXF flag is set as soon as the next message is shifted to the foreground buffer.

#### 16.4.7.4 Wake-Up Interrupt

A wake-up interrupt is generated if activity on the CAN bus occurs during MSCAN sleep or power-down mode.

**NOTE**

This interrupt can only occur if the MSCAN was in sleep mode (SLPRQ = 1 and SLPAK = 1) before entering power down mode, the wake-up option is enabled (WUPE = 1), and the wake-up interrupt is enabled (WUPIE = 1).

#### 16.4.7.5 Error Interrupt

An error interrupt is generated if an overrun of the receiver FIFO, error, warning, or bus-off condition occurrs. MSCAN Receiver Flag Register (CANRFLG) indicates one of the following conditions:

- **Overrun** — An overrun condition of the receiver FIFO as described in Section 16.4.2.3, "Receive Structures," occurred.
- **CAN Status Change** — The actual value of the transmit and receive error counters control the CAN bus state of the MSCAN. As soon as the error counters skip into a critical range (Tx/Rx-warning, Tx/Rx-error, bus-off) the MSCAN flags an error condition. The status change, which caused the error condition, is indicated by the TSTAT and RSTAT flags (see Section 16.3.2.5, "MSCAN Receiver Flag Register (CANRFLG)" and Section 16.3.2.6, "MSCAN Receiver Interrupt Enable Register (CANRIER)").

#### 16.4.7.6 Interrupt Acknowledge

Interrupts are directly associated with one or more status flags in either the MSCAN Receiver Flag Register (CANRFLG) or the MSCAN Transmitter Flag Register (CANTFLG). Interrupts are pending as long as one of the corresponding flags is set. The flags in CANRFLG and CANTFLG must be reset within the interrupt handler to handshake the interrupt. The flags are reset by writing a 1 to the corresponding bit position. A flag cannot be cleared if the respective condition prevails.

**NOTE**

It must be guaranteed that the CPU clears only the bit causing the current interrupt. For this reason, bit manipulation instructions (BSET) must not be used to clear interrupt flags. These instructions may cause accidental clearing of interrupt flags which are set after entering the current interrupt service routine.

## 16.5 Initialization/Application Information

### 16.5.1 MSCAN initialization

The procedure to initially start up the MSCAN module out of reset is as follows:

1. Assert CANE
2. Write to the configuration registers in initialization mode
3. Clear INITRQ to leave initialization mode

If the configuration of registers which are only writable in initialization mode shall be changed:

1. Bring the module into sleep mode by setting SLPRQ and awaiting SLPAK to assert after the CAN bus becomes idle.
2. Enter initialization mode: assert INITRQ and await INITAK
3. Write to the configuration registers in initialization mode
4. Clear INITRQ to leave initialization mode and continue

### 16.5.2 Bus-Off Recovery

The bus-off recovery is user configurable. The bus-off state can either be left automatically or on user request.

For reasons of backwards compatibility, the MSCAN defaults to automatic recovery after reset. In this case, the MSCAN will become error active again after counting 128 occurrences of 11 consecutive recessive bits on the CAN bus (see the Bosch CAN 2.0 A/B specification for details).

If the MSCAN is configured for user request (BORM set in MSCAN Control Register 1 (CANCTL1)), the recovery from bus-off starts after both independent events have become true:

- 128 occurrences of 11 consecutive recessive bits on the CAN bus have been monitored
- BOHOLD in MSCAN Miscellaneous Register (CANMISC) has been cleared by the user

These two events may occur in any order.

# Chapter 17
# Periodic Interrupt Timer (S12PIT24B8CV2)

**Table 17-1. Revision History**

| Revision Number | RevisionDate | Sections Affected | Description of Changes |
|---|---|---|---|
| V01.00 | 28 Apr 2005 | | - Initial Release. |
| V01.01 | 05 Jul 2005 | 17.6/17-674 | - Added application section.<br>- Removed table 1-1. |

## 17.1 Introduction

The period interrupt timer (PIT) is an array of 24-bit timers that can be used to trigger peripheral modules or raise periodic interrupts. Refer to Figure 17-1 for a simplified block diagram.

### 17.1.1 Glossary

| Acronyms and Abbreviations | |
|---|---|
| PIT | Periodic Interrupt Timer |
| ISR | Interrupt Service Routine |
| CCR | Condition Code Register |
| SoC | System on Chip |
| micro time bases | clock periods of the 16-bit timer modulus down-counters, which are generated by the 8-bit modulus down-counters. |

### 17.1.2 Features

The PIT includes these features:

- Eight timers implemented as modulus down-counters with independent time-out periods.
- Time-out periods selectable between 1 and $2^{24}$ bus clock cycles. Time-out equals m*n bus clock cycles with 1 <= m <= 256 and 1 <= n <= 65536.
- Timers that can be enabled individually.
- Eight time-out interrupts.
- Eight time-out trigger output signals available to trigger peripheral modules.
- Start of timer channels can be aligned to each other.

### 17.1.3 Modes of Operation

Refer to the device overview for a detailed explanation of the chip modes.

- Run mode

  This is the basic mode of operation.
- Wait mode

  PIT operation in wait mode is controlled by the PITSWAI bit located in the PITCFLMT register. In wait mode, if the bus clock is globally enabled and if the PITSWAI bit is clear, the PIT operates like in run mode. In wait mode, if the PITSWAI bit is set, the PIT module is stalled.
- Stop mode

  In full stop mode or pseudo stop mode, the PIT module is stalled.
- Freeze mode

  PIT operation in freeze mode is controlled by the PITFRZ bit located in the PITCFLMT register. In freeze mode, if the PITFRZ bit is clear, the PIT operates like in run mode. In freeze mode, if the PITFRZ bit is set, the PIT module is stalled.

### 17.1.4 Block Diagram

Figure 17-1 shows a block diagram of the PIT module.

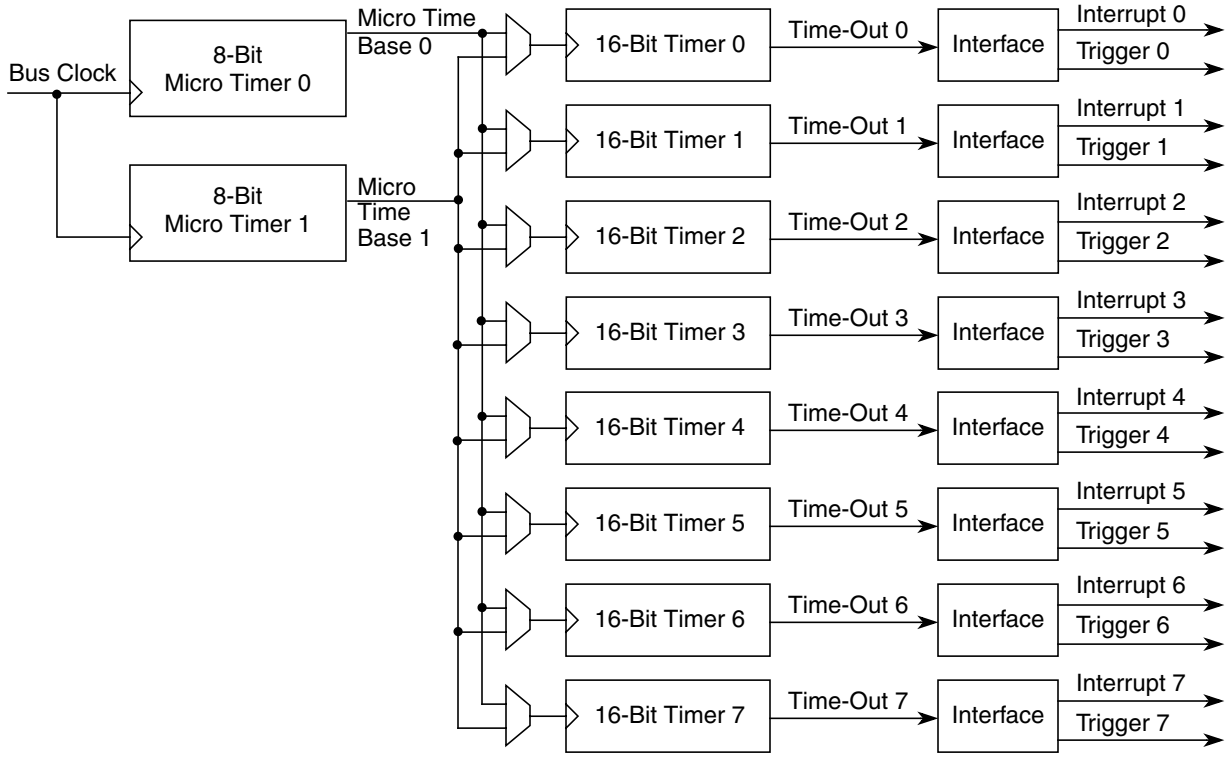



**Figure 17-1. PIT24B8C Block Diagram**

## 17.2 External Signal Description

The PIT module has no external pins.

## 17.3 Register Definition

This section consists of register descriptions in address order of the PIT. Each description includes a standard register diagram with an associated figure number. Details of register bit and field function follow the register diagrams, in bit order.

| Register Name | | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|---|
| 0x0000 PITCFLMT | R | PITE | PITSWAI | PITFRZ | 0 | 0 | 0 | 0 | 0 |
| | W | | | | | | | PFLMT1 | PFLMT0 |
| 0x0001 PITFLT | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | W | PFLT7 | PFLT6 | PFLT5 | PFLT4 | PFLT3 | PFLT2 | PFLT1 | PFLT0 |
| 0x0002 PITCE | R | PCE7 | PCE6 | PCE5 | PCE4 | PCE3 | PCE2 | PCE1 | PCE0 |
| | W | | | | | | | | |
| 0x0003 PITMUX | R | PMUX7 | PMUX6 | PMUX5 | PMUX4 | PMUX3 | PMUX2 | PMUX1 | PMUX0 |
| | W | | | | | | | | |
| 0x0004 PITINTE | R | PINTE7 | PINTE6 | PINTE5 | PINTE4 | PINTE3 | PINTE2 | PINTE1 | PINTE0 |
| | W | | | | | | | | |
| 0x0005 PITTF | R | PTF7 | PTF6 | PTF5 | PTF4 | PTF3 | PTF2 | PTF1 | PTF0 |
| | W | | | | | | | | |
| 0x0006 PITMTLD0 | R | PMTLD7 | PMTLD6 | PMTLD5 | PMTLD4 | PMTLD3 | PMTLD2 | PMTLD1 | PMTLD0 |
| | W | | | | | | | | |
| 0x0007 PITMTLD1 | R | PMTLD7 | PMTLD6 | PMTLD5 | PMTLD4 | PMTLD3 | PMTLD2 | PMTLD1 | PMTLD0 |
| | W | | | | | | | | |
| 0x0008 PITLD0 (High) | R | PLD15 | PLD14 | PLD13 | PLD12 | PLD11 | PLD10 | PLD9 | PLD8 |
| | W | | | | | | | | |
| 0x0009 PITLD0 (Low) | R | PLD7 | PLD6 | PLD5 | PLD4 | PLD3 | PLD2 | PLD1 | PLD0 |
| | W | | | | | | | | |
| 0x000A PITCNT0 (High) | R | PCNT15 | PCNT14 | PCNT13 | PCNT12 | PCNT11 | PCNT10 | PCNT9 | PCNT8 |
| | W | | | | | | | | |
| 0x000B PITCNT0 (Low) | R | PCNT7 | PCNT6 | PCNT5 | PCNT4 | PCNT3 | PCNT2 | PCNT1 | PCNT0 |
| | W | | | | | | | | |
| 0x000C PITLD1 (High) | R | PLD15 | PLD14 | PLD13 | PLD12 | PLD11 | PLD10 | PLD9 | PLD8 |
| | W | | | | | | | | |

= Unimplemented or Reserved

**Figure 17-2. PIT Register Summary (Sheet 1 of 3)**

| Register Name | | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|---|
| 0x000D PITLD1 (Low) | R<br>W | PLD7 | PLD6 | PLD5 | PLD4 | PLD3 | PLD2 | PLD1 | PLD0 |
| 0x000E PITCNT1 (High) | R<br>W | PCNT15 | PCNT14 | PCNT13 | PCNT12 | PCNT11 | PCNT10 | PCNT9 | PCNT8 |
| 0x000F PITCNT1 (Low) | R<br>W | PCNT7 | PCNT6 | PCNT5 | PCNT4 | PCNT3 | PCNT2 | PCNT1 | PCNT0 |
| 0x0010 PITLD2 (High) | R<br>W | PLD15 | PLD14 | PLD13 | PLD12 | PLD11 | PLD10 | PLD9 | PLD8 |
| 0x0011 PITLD2 (Low) | R<br>W | PLD7 | PLD6 | PLD5 | PLD4 | PLD3 | PLD2 | PLD1 | PLD0 |
| 0x0012 PITCNT2 (High) | R<br>W | PCNT15 | PCNT14 | PCNT13 | PCNT12 | PCNT11 | PCNT10 | PCNT9 | PCNT8 |
| 0x0013 PITCNT2 (Low) | R<br>W | PCNT7 | PCNT6 | PCNT5 | PCNT4 | PCNT3 | PCNT2 | PCNT1 | PCNT0 |
| 0x0014 PITLD3 (High) | R<br>W | PLD15 | PLD14 | PLD13 | PLD12 | PLD11 | PLD10 | PLD9 | PLD8 |
| 0x0015 PITLD3 (Low) | R<br>W | PLD7 | PLD6 | PLD5 | PLD4 | PLD3 | PLD2 | PLD1 | PLD0 |
| 0x0016 PITCNT3 (High) | R<br>W | PCNT15 | PCNT14 | PCNT13 | PCNT12 | PCNT11 | PCNT10 | PCNT9 | PCNT8 |
| 0x0017 PITCNT3 (Low) | R<br>W | PCNT7 | PCNT6 | PCNT5 | PCNT4 | PCNT3 | PCNT2 | PCNT1 | PCNT0 |
| 0x0018 PITLD4 (High) | R<br>W | PLD15 | PLD14 | PLD13 | PLD12 | PLD11 | PLD10 | PLD9 | PLD8 |
| 0x0019 PITLD4 (Low) | R<br>W | PLD7 | PLD6 | PLD5 | PLD4 | PLD3 | PLD2 | PLD1 | PLD0 |
| 0x001A PITCNT4 (High) | R<br>W | PCNT15 | PCNT14 | PCNT13 | PCNT12 | PCNT11 | PCNT10 | PCNT9 | PCNT8 |
| 0x001B PITCNT4 (Low) | R<br>W | PCNT7 | PCNT6 | PCNT5 | PCNT4 | PCNT3 | PCNT2 | PCNT1 | PCNT0 |

= Unimplemented or Reserved

**Figure 17-2. PIT Register Summary (Sheet 2 of 3)**

| Register Name | | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|---|
| 0x001C PITLD5 (High) | R<br>W | PLD15 | PLD14 | PLD13 | PLD12 | PLD11 | PLD10 | PLD9 | PLD8 |
| 0x001D PITLD5 (Low) | R<br>W | PLD7 | PLD6 | PLD5 | PLD4 | PLD3 | PLD2 | PLD1 | PLD0 |
| 0x001E PITCNT5 (High) | R<br>W | PCNT15 | PCNT14 | PCNT13 | PCNT12 | PCNT11 | PCNT10 | PCNT9 | PCNT8 |
| 0x001F PITCNT5 (Low) | R<br>W | PCNT7 | PCNT6 | PCNT5 | PCNT4 | PCNT3 | PCNT2 | PCNT1 | PCNT0 |
| 0x0020 PITLD6 (High) | R<br>W | PLD15 | PLD14 | PLD13 | PLD12 | PLD11 | PLD10 | PLD9 | PLD8 |
| 0x0021 PITLD6 (Low) | R<br>W | PLD7 | PLD6 | PLD5 | PLD4 | PLD3 | PLD2 | PLD1 | PLD0 |
| 0x0022 PITCNT6 (High) | R<br>W | PCNT15 | PCNT14 | PCNT13 | PCNT12 | PCNT11 | PCNT10 | PCNT9 | PCNT8 |
| 0x0023 PITCNT6 (Low) | R<br>W | PCNT7 | PCNT6 | PCNT5 | PCNT4 | PCNT3 | PCNT2 | PCNT1 | PCNT0 |
| 0x0024 PITLD7 (High) | R<br>W | PLD15 | PLD14 | PLD13 | PLD12 | PLD11 | PLD10 | PLD9 | PLD8 |
| 0x0025 PITLD7 (Low) | R<br>W | PLD7 | PLD6 | PLD5 | PLD4 | PLD3 | PLD2 | PLD1 | PLD0 |
| 0x0026 PITCNT7 (High) | R<br>W | PCNT15 | PCNT14 | PCNT13 | PCNT12 | PCNT11 | PCNT10 | PCNT9 | PCNT8 |
| 0x0027 PITCNT7 (Low) | R<br>W | PCNT7 | PCNT6 | PCNT5 | PCNT4 | PCNT3 | PCNT2 | PCNT1 | PCNT0 |

= Unimplemented or Reserved

**Figure 17-2. PIT Register Summary (Sheet 3 of 3)**

### 17.3.0.1 PIT Control and Force Load Micro Timer Register (PITCFLMT)

Module Base + 0x0000

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | PITE | PITSWAI | PITFRZ | 0 | 0 | 0 | 0 | 0 |
| W | | | | | | | PFLMT1 | PFLMT0 |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented or Reserved

**Figure 17-3. PIT Control and Force Load Micro Timer Register (PITCFLMT)**

Read: Anytime

Write: Anytime; writes to the reserved bits have no effect

**Table 17-2. PITCFLMT Field Descriptions**

| Field | Description |
|---|---|
| 7<br>PITE | **PIT Module Enable Bit** — This bit enables the PIT module. If PITE is cleared, the PIT module is disabled and flag bits in the PITTF register are cleared. When PITE is set, individually enabled timers (PCE set) start down-counting with the corresponding load register values.<br>0 PIT disabled (lower power consumption).<br>1 PIT is enabled. |
| 6<br>PITSWAI | **PIT Stop in Wait Mode Bit** — This bit is used for power conservation while in wait mode.<br>0 PIT operates normally in wait mode<br>1 PIT clock generation stops and freezes the PIT module when in wait mode |
| 5<br>PITFRZ | **PIT Counter Freeze while in Freeze Mode Bit** — When during debugging a breakpoint (freeze mode) is encountered it is useful in many cases to freeze the PIT counters to avoid e.g. interrupt generation. The PITFRZ bit controls the PIT operation while in freeze mode.<br>0 PIT operates normally in freeze mode<br>1 PIT counters are stalled when in freeze mode |
| 1:0<br>PFLMT[1:0] | **PIT Force Load Bits for Micro Timer 1:0** — These bits have only an effect if the corresponding micro timer is active and if the PIT module is enabled (PITE set). Writing a one into a PFLMT bit loads the corresponding 8-bit micro timer load register into the 8-bit micro timer down-counter. Writing a zero has no effect. Reading these bits will always return zero.<br>**Note:** A micro timer force load affects all timer channels that use the corresponding micro time base. |

### 17.3.0.2 PIT Force Load Timer Register (PITFLT)

Module Base + 0x0001

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| W | PFLT7 | PFLT6 | PFLT5 | PFLT4 | PFLT3 | PFLT2 | PFLT1 | PFLT0 |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 17-4. PIT Force Load Timer Register (PITFLT)**

Read: Anytime

Write: Anytime

**Table 17-3. PITFLT Field Descriptions**

| Field | Description |
|---|---|
| 7:0<br>PFLT[7:0] | **PIT Force Load Bits for Timer 7-0** — These bits have only an effect if the corresponding timer channel (PCE set) is enabled and if the PIT module is enabled (PITE set). Writing a one into a PFLT bit loads the corresponding 16-bit timer load register into the 16-bit timer down-counter. Writing a zero has no effect. Reading these bits will always return zero. |

### 17.3.0.3 PIT Channel Enable Register (PITCE)

Module Base + 0x0002

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | PCE7 | PCE6 | PCE5 | PCE4 | PCE3 | PCE2 | PCE1 | PCE0 |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 17-5. PIT Channel Enable Register (PITCE)**

Read: Anytime

Write: Anytime

**Table 17-4. PITCE Field Descriptions**

| Field | Description |
|---|---|
| 7:0<br>PCE[7:0] | **PIT Enable Bits for Timer Channel 7:0** — These bits enable the PIT channels 7-0. If PCE is cleared, the PIT channel is disabled and the corresponding flag bit in the PITTF register is cleared. When PCE is set, and if the PIT module is enabled (PITE = 1) the 16-bit timer counter is loaded with the start count value and starts down-counting.<br>0 The corresponding PIT channel is disabled.<br>1 The corresponding PIT channel is enabled. |

### 17.3.0.4 PIT Multiplex Register (PITMUX)

Module Base + 0x0003

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | PMUX7 | PMUX6 | PMUX5 | PMUX4 | PMUX3 | PMUX2 | PMUX1 | PMUX0 |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 17-6. PIT Multiplex Register (PITMUX)**

Read: Anytime

Write: Anytime

**Table 17-5. PITMUX Field Descriptions**

| Field | Description |
|---|---|
| 7:0<br>PMUX[7:0] | **PIT Multiplex Bits for Timer Channel 7:0** — These bits select if the corresponding 16-bit timer is connected to micro time base 1 or 0. If PMUX is modified, the corresponding 16-bit timer is immediately switched to the other micro time base.<br>0 The corresponding 16-bit timer counts with micro time base 0.<br>1 The corresponding 16-bit timer counts with micro time base 1. |

### 17.3.0.5 PIT Interrupt Enable Register (PITINTE)

Module Base + 0x0004

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | PINTE7 | PINTE6 | PINTE5 | PINTE4 | PINTE3 | PINTE2 | PINTE1 | PINTE0 |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 17-7. PIT Interrupt Enable Register (PITINTE)**

Read: Anytime

Write: Anytime

**Table 17-6. PITINTE Field Descriptions**

| Field | Description |
|---|---|
| 7:0<br>PINTE[7:0] | **PIT Time-out Interrupt Enable Bits for Timer Channel 7:0** — These bits enable an interrupt service request whenever the time-out flag PTF of the corresponding PIT channel is set. When an interrupt is pending (PTF set) enabling the interrupt will immediately cause an interrupt. To avoid this, the corresponding PTF flag has to be cleared first.<br>0 Interrupt of the corresponding PIT channel is disabled.<br>1 Interrupt of the corresponding PIT channel is enabled. |

### 17.3.0.6 PIT Time-Out Flag Register (PITTF)

Module Base + 0x0005

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | PTF7 | PTF6 | PTF5 | PTF4 | PTF3 | PTF2 | PTF1 | PTF0 |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 17-8. PIT Time-Out Flag Register (PITTF)**

Read: Anytime

Write: Anytime (write to clear)

**Table 17-7. PITTF Field Descriptions**

| Field | Description |
|---|---|
| 7:0<br>PTF[7:0] | **PIT Time-out Flag Bits for Timer Channel 7:0** — PTF is set when the corresponding 16-bit timer modulus down-counter and the selected 8-bit micro timer modulus down-counter have counted to zero. The flag can be cleared by writing a one to the flag bit. Writing a zero has no effect. If flag clearing by writing a one and flag setting happen in the same bus clock cycle, the flag remains set. The flag bits are cleared if the PIT module is disabled or if the corresponding timer channel is disabled.<br>0 Time-out of the corresponding PIT channel has not yet occurred.<br>1 Time-out of the corresponding PIT channel has occurred. |

## 17.3.0.7 PIT Micro Timer Load Register 0 to 1 (PITMTLD0–1)

Module Base + 0x0006

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | PMTLD7 | PMTLD6 | PMTLD5 | PMTLD4 | PMTLD3 | PMTLD2 | PMTLD1 | PMTLD0 |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 17-9. PIT Micro Timer Load Register 0 (PITMTLD0)**

Module Base + 0x0007

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | PMTLD7 | PMTLD6 | PMTLD5 | PMTLD4 | PMTLD3 | PMTLD2 | PMTLD1 | PMTLD0 |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 17-10. PIT Micro Timer Load Register 1 (PITMTLD1)**

Read: Anytime

Write: Anytime

**Table 17-8. PITMTLD0–1 Field Descriptions**

| Field | Description |
|---|---|
| 7:0<br>PMTLD[7:0] | **PIT Micro Timer Load Bits 7:0** — These bits set the 8-bit modulus down-counter load value of the micro timers. Writing a new value into the PITMTLD register will not restart the timer. When the micro timer has counted down to zero, the PMTLD register value will be loaded. The PFLMT bits in the PITCFLMT register can be used to immediately update the count register with the new value if an immediate load is desired. |

## 17.3.0.8 PIT Load Register 0 to 7 (PITLD0–7)

Module Base + 0x0008, 0x0009

| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | PLD15 | PLD14 | PLD13 | PLD12 | PLD11 | PLD10 | PLD9 | PLD8 | PLD7 | PLD6 | PLD5 | PLD4 | PLD3 | PLD2 | PLD1 | PLD0 |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 17-11. PIT Load Register 0 (PITLD0)**

Module Base + 0x000C, 0x000D

| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | PLD15 | PLD14 | PLD13 | PLD12 | PLD11 | PLD10 | PLD9 | PLD8 | PLD7 | PLD6 | PLD5 | PLD4 | PLD3 | PLD2 | PLD1 | PLD0 |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 17-12. PIT Load Register 1 (PITLD1)**

Module Base + 0x0010, 0x0011

| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | PLD15 | PLD14 | PLD13 | PLD12 | PLD11 | PLD10 | PLD9 | PLD8 | PLD7 | PLD6 | PLD5 | PLD4 | PLD3 | PLD2 | PLD1 | PLD0 |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 17-13. PIT Load Register 2 (PITLD2)**

Module Base + 0x0014, 0x0015

| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | PLD15 | PLD14 | PLD13 | PLD12 | PLD11 | PLD10 | PLD9 | PLD8 | PLD7 | PLD6 | PLD5 | PLD4 | PLD3 | PLD2 | PLD1 | PLD0 |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 17-14. PIT Load Register 3 (PITLD3)**

Module Base + 0x0018, 0x0019

| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | PLD15 | PLD14 | PLD13 | PLD12 | PLD11 | PLD10 | PLD9 | PLD8 | PLD7 | PLD6 | PLD5 | PLD4 | PLD3 | PLD2 | PLD1 | PLD0 |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 17-15. PIT Load Register 4 (PITLD4)**

Module Base + 0x001C, 0x001D

| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | PLD15 | PLD14 | PLD13 | PLD12 | PLD11 | PLD10 | PLD9 | PLD8 | PLD7 | PLD6 | PLD5 | PLD4 | PLD3 | PLD2 | PLD1 | PLD0 |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 17-16. PIT Load Register 5 (PITLD5)**

Module Base + 0x0020, 0x0021

| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | PLD15 | PLD14 | PLD13 | PLD12 | PLD11 | PLD10 | PLD9 | PLD8 | PLD7 | PLD6 | PLD5 | PLD4 | PLD3 | PLD2 | PLD1 | PLD0 |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 17-17. PIT Load Register 6 (PITLD6)**

Module Base + 0x0024, 0x0025

| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | PLD15 | PLD14 | PLD13 | PLD12 | PLD11 | PLD10 | PLD9 | PLD8 | PLD7 | PLD6 | PLD5 | PLD4 | PLD3 | PLD2 | PLD1 | PLD0 |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 17-18. PIT Load Register 7 (PITLD7)**

Read: Anytime

Write: Anytime

**Table 17-9. PITLD0–7 Field Descriptions**

| Field | Description |
|---|---|
| 15:0<br>PLD[15:0] | **PIT Load Bits 15:0** — These bits set the 16-bit modulus down-counter load value. Writing a new value into the PITLD register must be a 16-bit access, to ensure data consistency. It will not restart the timer. When the timer has counted down to zero the PTF time-out flag will be set and the register value will be loaded. The PFLT bits in the PITFLT register can be used to immediately update the count register with the new value if an immediate load is desired. |

## 17.3.0.9 PIT Count Register 0 to 7 (PITCNT0–7)

Module Base + 0x000A, 0x000B

| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | PCNT 15 | PCNT 14 | PCNT 13 | PCNT 12 | PCNT 11 | PCNT 10 | PCN T9 | PCN T8 | PCN T7 | PCN T6 | PCN T5 | PCN T4 | PCN T3 | PCN T2 | PCN T1 | PCN T0 |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 17-19. PIT Count Register 0 (PITCNT0)**

Module Base + 0x000E, 0x000F

| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | PCNT 15 | PCNT 14 | PCNT 13 | PCNT 12 | PCNT 11 | PCNT 10 | PCN T9 | PCN T8 | PCN T7 | PCN T6 | PCN T5 | PCN T4 | PCN T3 | PCN T2 | PCN T1 | PCN T0 |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 17-20. PIT Count Register 1 (PITCNT1)**

Module Base + 0x0012, 0x0013

| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | PCNT 15 | PCNT 14 | PCNT 13 | PCNT 12 | PCNT 11 | PCNT 10 | PCN T9 | PCN T8 | PCN T7 | PCN T6 | PCN T5 | PCN T4 | PCN T3 | PCN T2 | PCN T1 | PCN T0 |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 17-21. PIT Count Register 2 (PITCNT2)**

Module Base + 0x0016, 0x0017

| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | PCNT 15 | PCNT 14 | PCNT 13 | PCNT 12 | PCNT 11 | PCNT 10 | PCN T9 | PCN T8 | PCN T7 | PCN T6 | PCN T5 | PCN T4 | PCN T3 | PCN T2 | PCN T1 | PCN T0 |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 17-22. PIT Count Register 3 (PITCNT3)**

Module Base + 0x001A, 0x001B

| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | PCNT 15 | PCNT 14 | PCNT 13 | PCNT 12 | PCNT 11 | PCNT 10 | PCN T9 | PCN T8 | PCN T7 | PCN T6 | PCN T5 | PCN T4 | PCN T3 | PCN T2 | PCN T1 | PCN T0 |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 17-23. PIT Count Register 4 (PITCNT4)**

Module Base + 0x001E, 0x001F

| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | PCNT 15 | PCNT 14 | PCNT 13 | PCNT 12 | PCNT 11 | PCNT 10 | PCN T9 | PCN T8 | PCN T7 | PCN T6 | PCN T5 | PCN T4 | PCN T3 | PCN T2 | PCN T1 | PCN T0 |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 17-24. PIT Count Register 5 (PITCNT5)**

Module Base + 0x0022, 0x0023

| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | PCNT 15 | PCNT 14 | PCNT 13 | PCNT 12 | PCNT 11 | PCNT 10 | PCN T9 | PCN T8 | PCN T7 | PCN T6 | PCN T5 | PCN T4 | PCN T3 | PCN T2 | PCN T1 | PCN T0 |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 17-25. PIT Count Register 6 (PITCNT6)**

Module Base + 0x0026, 0x0027

| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | PCNT 15 | PCNT 14 | PCNT 13 | PCNT 12 | PCNT 11 | PCNT 10 | PCN T9 | PCN T8 | PCN T7 | PCN T6 | PCN T5 | PCN T4 | PCN T3 | PCN T2 | PCN T1 | PCN T0 |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 17-26. PIT Count Register 7 (PITCNT7)**

Read: Anytime

Write: Has no meaning or effect

**Table 17-10. PITCNT0–7 Field Descriptions**

| Field | Description |
|---|---|
| 15:0<br>PCNT[15:0] | **PIT Count Bits 15-0** — These bits represent the current 16-bit modulus down-counter value. The read access for the count register must take place in one clock cycle as a 16-bit access. |

## 17.4 Functional Description

Figure 17-27 shows a detailed block diagram of the PIT module. The main parts of the PIT are status, control and data registers, two 8-bit down-counters, eight 16-bit down-counters and an interrupt/trigger interface.

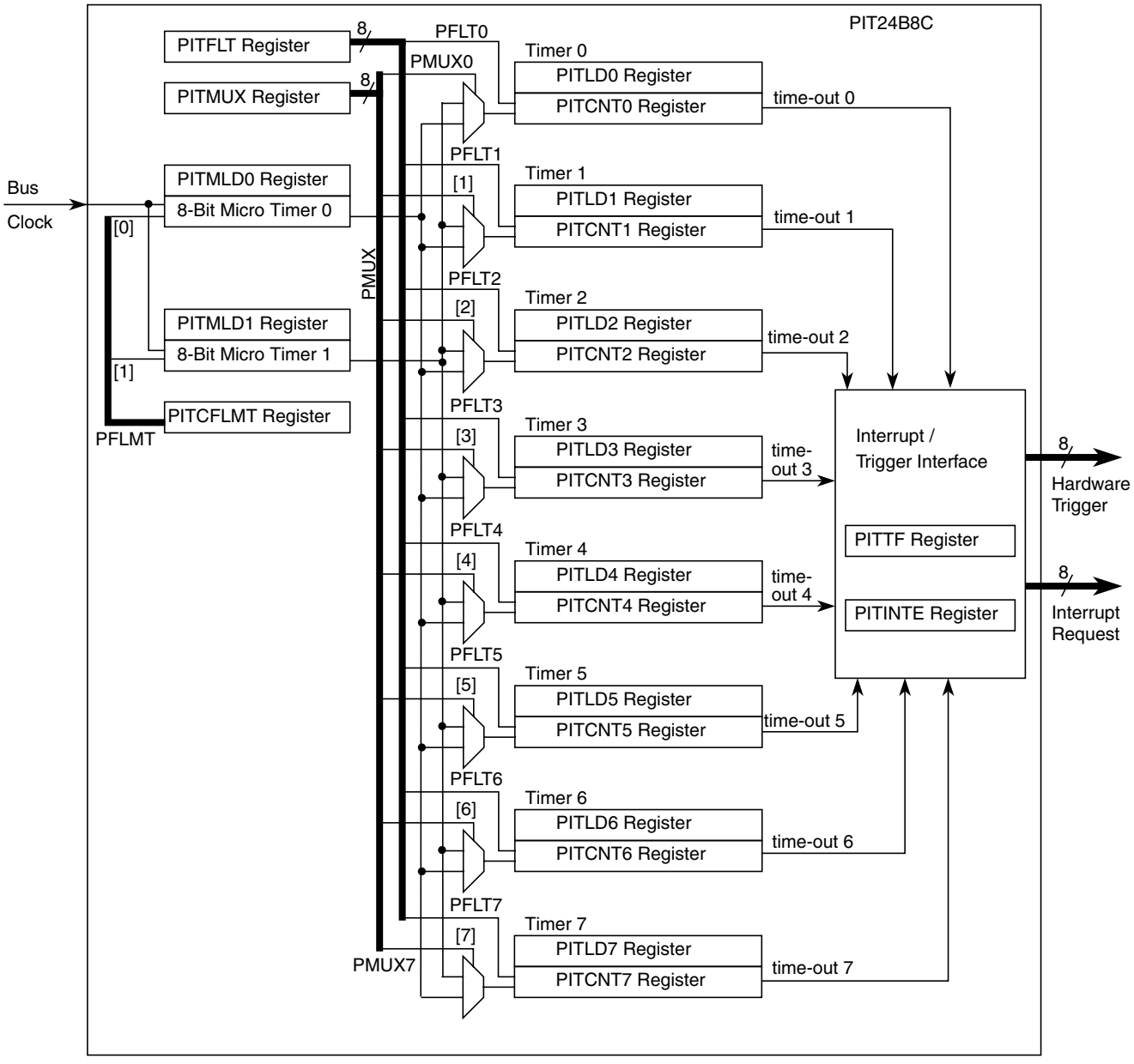



**Figure 17-27. PIT24B8C Detailed Block Diagram**

### 17.4.1 Timer

As shown in Figure 17-1 and Figure 17-27, the 24-bit timers are built in a two-stage architecture with eight 16-bit modulus down-counters and two 8-bit modulus down-counters. The 16-bit timers are clocked with two selectable micro time bases which are generated with 8-bit modulus down-counters. Each 16-bit timer is connected to micro time base 0 or 1 via the PMUX[7:0] bit setting in the PIT Multiplex (PITMUX) register.

A timer channel is enabled if the module enable bit PITE in the PIT control and force load micro timer (PITCFLMT) register is set and if the corresponding PCE bit in the PIT channel enable (PITCE) register is set. Two 8-bit modulus down-counters are used to generate two micro time bases. As soon as a micro time base is selected for an enabled timer channel, the corresponding micro timer modulus down-counter will load its start value as specified in the PITMTLD0 or PITMTLD1 register and will start down-counting. Whenever the micro timer down-counter has counted to zero the PITMTLD register is reloaded and the connected 16-bit modulus down-counters count one cycle.

Whenever a 16-bit timer counter and the connected 8-bit micro timer counter have counted to zero, the PITLD register is reloaded and the corresponding time-out flag PTF in the PIT time-out flag (PITTF) register is set, as shown in Figure 17-28. The time-out period is a function of the timer load (PITLD) and micro timer load (PITMTLD) registers and the bus clock $f_{BUS}$:

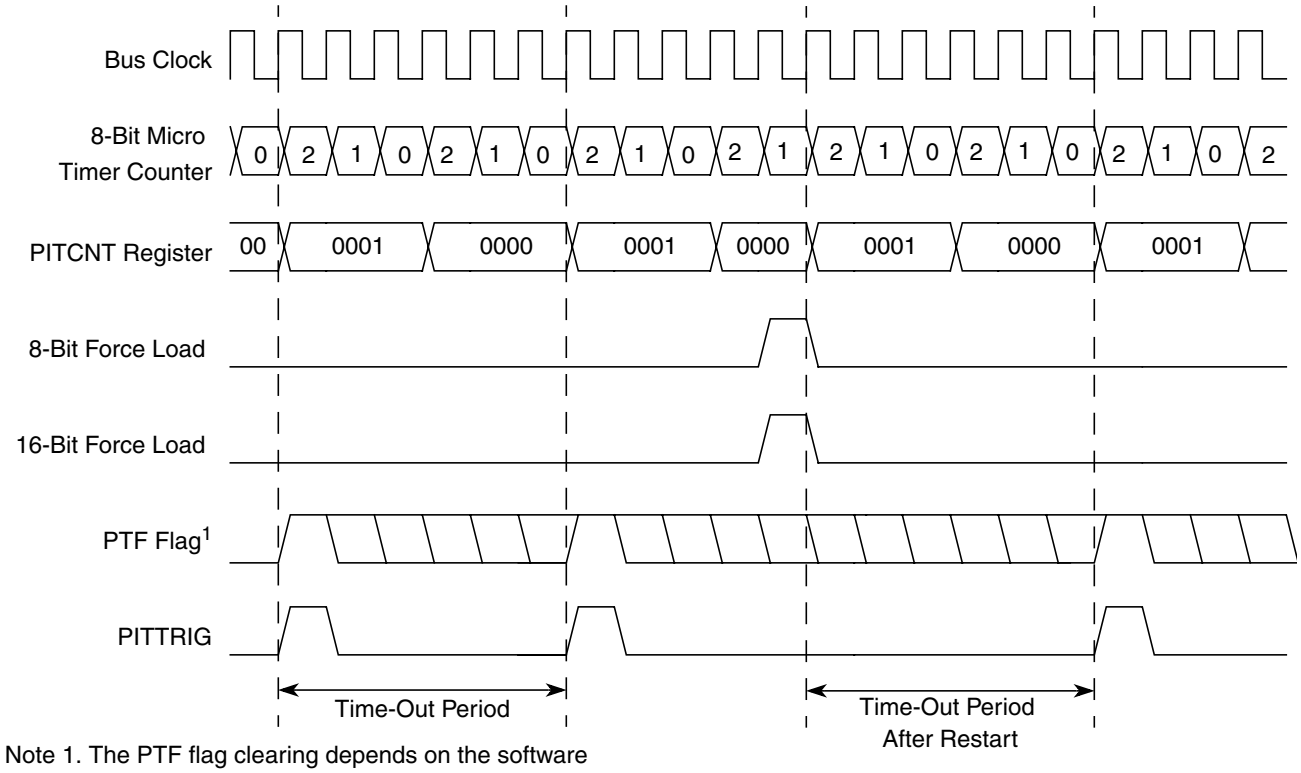
time-out period = (PITMTLD + 1) * (PITLD + 1) / $f_{BUS}$.

For example, for a 40 MHz bus clock, the maximum time-out period equals:

256 * 65536 * 25 ns = 419.43 ms.

The current 16-bit modulus down-counter value can be read via the PITCNT register. The micro timer down-counter values cannot be read.

The 8-bit micro timers can individually be restarted by writing a one to the corresponding force load micro timer PFLMT bits in the PIT control and force load micro timer (PITCFLMT) register. The 16-bit timers can individually be restarted by writing a one to the corresponding force load timer PFLT bits in the PIT forceload timer (PITFLT) register. If desired, any group of timers and micro timers can be restarted at the same time by using one 16-bit write to the adjacent PITCFLMT and PITFLT registers with the relevant bits set, as shown in Figure 17-28.




Note 1. The PTF flag clearing depends on the software

**Figure 17-28. PIT Trigger and Flag Signal Timing**

## 17.4.2 Interrupt Interface

Each time-out event can be used to trigger an interrupt service request. For each timer channel, an individual bit PINTE in the PIT interrupt enable (PITINTE) register exists to enable this feature. If PINTE is set, an interrupt service is requested whenever the corresponding time-out flag PTF in the PIT time-out flag (PITTF) register is set. The flag can be cleared by writing a one to the flag bit.

**NOTE**

Be careful when resetting the PITE, PINTE or PITCE bits in case of pending PIT interrupt requests, to avoid spurious interrupt requests.

## 17.4.3 Hardware Trigger

The PIT module contains eight hardware trigger signal lines PITTRIG[7:0], one for each timer channel. These signals can be connected on SoC level to peripheral modules enabling e.g. periodic ATD conversion (please refer to the device overview for the mapping of PITTRIG[7:0] signals to peripheral modules).

Whenever a timer channel time-out is reached, the corresponding PTF flag is set and the corresponding trigger signal PITTRIG triggers a rising edge. The trigger feature requires a minimum time-out period of two bus clock cycles because the trigger is asserted high for at least one bus clock cycle. For load register values PITLD = 0x0001 and PITMTLD = 0x0002 the flag setting, trigger timing and a restart with force load is shown in Figure 17-28.

## 17.5 Initialization

### 17.5.1 Startup

Set the configuration registers before the PITE bit in the PITCFLMT register is set. Before PITE is set, the configuration registers can be written in arbitrary order.

### 17.5.2 Shutdown

When the PITCE register bits, the PITINTE register bits or the PITE bit in the PITCFLMT register are cleared, the corresponding PIT interrupt flags are cleared. In case of a pending PIT interrupt request, a spurious interrupt can be generated. Two strategies, which avoid spurious interrupts, are recommended:

1. Reset the PIT interrupt flags only in an ISR. When entering the ISR, the I mask bit in the CCR is set automatically. The I mask bit must not be cleared before the PIT interrupt flags are cleared.
2. After setting the I mask bit with the SEI instruction, the PIT interrupt flags can be cleared. Then clear the I mask bit with the CLI instruction to re-enable interrupts.

### 17.5.3 Flag Clearing

A flag is cleared by writing a one to the flag bit. Always use store or move instructions to write a one in certain bit positions. Do not use the BSET instructions. Do not use any C-constructs that compile to BSET instructions. “BSET flag_register, #mask” must not be used for flag clearing because BSET is a read-modify-write instruction which writes back the “bit-wise or” of the flag_register and the mask into the flag_register. BSET would clear all flag bits that were set, independent from the mask.

For example, to clear flag bit 0 use: MOVB #$01,PITTF.

## 17.6 Application Information

To get started quickly with the PIT24B8C module this section provides a small code example how to use the block. Please note that the example provided is only one specific case out of the possible configurations and implementations.

Functionality: Generate an PIT interrupt on channel 0 every 500 PIT clock cycles.

```
                ORG     CODESTART               ; place the program into specific
                                                ; range (to be selected)
                LDS     RAMEND                  ; load stack pointer to top of RAM
                MOVW    #CH0_ISR,VEC_PIT_CH0    ; Change value of channel 0 ISR adr

; ******************** Start PIT Initialization ***************************************************

                CLR     PITCFLMT                ; disable PIT
                MOVB    #$01,PITCE              ; enable timer channel 0
                CLR     PITMUX                  ; ch0 connected to micro timer 0
                MOVB    #$63,PITMTLD0           ; micro time base 0 equals 100 clock cycles
                MOVW    #$0004,PITLD0           ; time base 0 eq. 5 micro time bases 0 =5*100 = 500
```

```
                MOVB    #$01,PITINTE            ; enable interupt channel 0
                MOVB    #$80,PITCFLMT           ; enable PIT
                CLI                             ; clear Interupt disable Mask bit

;******************** Main Program ************************************************************

MAIN:           BRA *                           ; loop until interrupt

;******************** Channel 0 Interupt Routine **********************************************

CH0_ISR:        LDAA    PITTF                   ; 8 bit read of PIT time out flags
                MOVB    #$01,PITTF              ; clear PIT channel 0 time out flag
                RTI                             ; return to MAIN
```

# Chapter 18
# Periodic Interrupt Timer (S12PIT24B4CV2)

**Table 18-1. Revision History**

| Revision Number | Revision Date | Sections Affected | Description of Changes |
|---|---|---|---|
| V01.00 | 28 Apr 2005 | | - Initial Release |
| V01.01 | 05 Jul 2005 | 18.6/18-690 | - Added application section.<br>- Removed table 1-1 |

## 18.1 Introduction

The period interrupt timer (PIT) is an array of 24-bit timers that can be used to trigger peripheral modules or raise periodic interrupts. Refer to Figure 18-1 for a simplified block diagram.

### 18.1.1 Glossary

| Acronyms and Abbreviations | |
|---|---|
| PIT | Periodic Interrupt Timer |
| ISR | Interrupt Service Routine |
| CCR | Condition Code Register |
| SoC | System on Chip |
| micro time bases | clock periods of the 16-bit timer modulus down-counters, which are generated by the 8-bit modulus down-counters. |

### 18.1.2 Features

The PIT includes these features:

- Four timers implemented as modulus down-counters with independent time-out periods.
- Time-out periods selectable between 1 and $2^{24}$ bus clock cycles. Time-out equals m*n bus clock cycles with 1 <= m <= 256 and 1 <= n <= 65536.
- Timers that can be enabled individually.
- Four time-out interrupts.
- Four time-out trigger output signals available to trigger peripheral modules.
- Start of timer channels can be aligned to each other.

### 18.1.3 Modes of Operation

Refer to the device overview for a detailed explanation of the chip modes.

- Run mode

  This is the basic mode of operation.
- Wait mode

  PIT operation in wait mode is controlled by the PITSWAI bit located in the PITCFLMT register. In wait mode, if the bus clock is globally enabled and if the PITSWAI bit is clear, the PIT operates like in run mode. In wait mode, if the PITSWAI bit is set, the PIT module is stalled.
- Stop mode

  In full stop mode or pseudo stop mode, the PIT module is stalled.
- Freeze mode

  PIT operation in freeze mode is controlled by the PITFRZ bit located in the PITCFLMT register. In freeze mode, if the PITFRZ bit is clear, the PIT operates like in run mode. In freeze mode, if the PITFRZ bit is set, the PIT module is stalled.

### 18.1.4 Block Diagram

Figure 18-1 shows a block diagram of the PIT module.

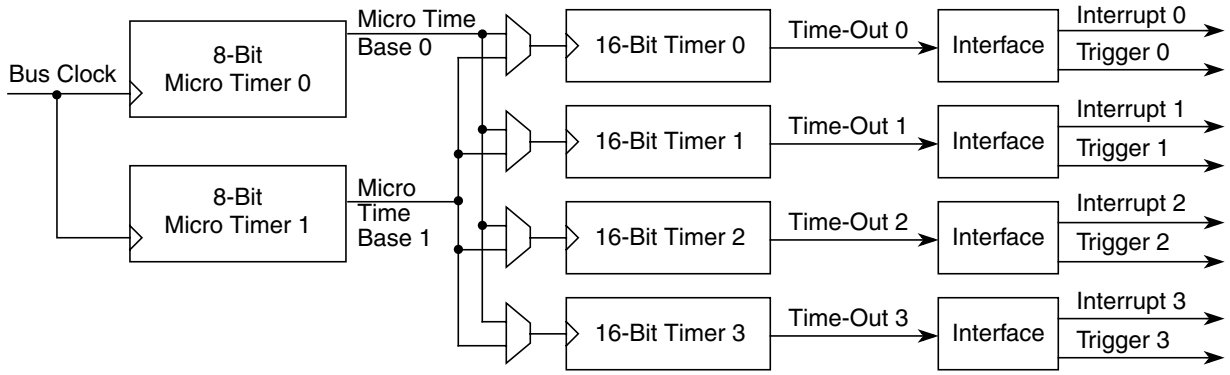



**Figure 18-1. PIT24B4C Block Diagram**

## 18.2 External Signal Description

The PIT module has no external pins.

## 18.3 Register Definition

This section consists of register descriptions in address order of the PIT. Each description includes a standard register diagram with an associated figure number. Details of register bit and field function follow the register diagrams, in bit order.

| Register Name | | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|---|
| 0x0000 PITCFLMT | R | PITE | PITSWAI | PITFRZ | 0 | 0 | 0 | 0 | 0 |
| | W | | | | | | | PFLMT1 | PFLMT0 |
| 0x0001 PITFLT | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | W | | | | | PFLT3 | PFLT2 | PFLT1 | PFLT0 |
| 0x0002 PITCE | R | 0 | 0 | 0 | 0 | PCE3 | PCE2 | PCE1 | PCE0 |
| | W | | | | | | | | |
| 0x0003 PITMUX | R | 0 | 0 | 0 | 0 | PMUX3 | PMUX2 | PMUX1 | PMUX0 |
| | W | | | | | | | | |
| 0x0004 PITINTE | R | 0 | 0 | 0 | 0 | PINTE3 | PINTE2 | PINTE1 | PINTE0 |
| | W | | | | | | | | |
| 0x0005 PITTF | R | 0 | 0 | 0 | 0 | PTF3 | PTF2 | PTF1 | PTF0 |
| | W | | | | | | | | |
| 0x0006 PITMTLD0 | R | PMTLD7 | PMTLD6 | PMTLD5 | PMTLD4 | PMTLD3 | PMTLD2 | PMTLD1 | PMTLD0 |
| | W | | | | | | | | |
| 0x0007 PITMTLD1 | R | PMTLD7 | PMTLD6 | PMTLD5 | PMTLD4 | PMTLD3 | PMTLD2 | PMTLD1 | PMTLD0 |
| | W | | | | | | | | |
| 0x0008 PITLD0 (High) | R | PLD15 | PLD14 | PLD13 | PLD12 | PLD11 | PLD10 | PLD9 | PLD8 |
| | W | | | | | | | | |
| 0x0009 PITLD0 (Low) | R | PLD7 | PLD6 | PLD5 | PLD4 | PLD3 | PLD2 | PLD1 | PLD0 |
| | W | | | | | | | | |
| 0x000A PITCNT0 (High) | R | PCNT15 | PCNT14 | PCNT13 | PCNT12 | PCNT11 | PCNT10 | PCNT9 | PCNT8 |
| | W | | | | | | | | |
| 0x000B PITCNT0 (Low) | R | PCNT7 | PCNT6 | PCNT5 | PCNT4 | PCNT3 | PCNT2 | PCNT1 | PCNT0 |
| | W | | | | | | | | |
| 0x000C PITLD1 (High) | R | PLD15 | PLD14 | PLD13 | PLD12 | PLD11 | PLD10 | PLD9 | PLD8 |
| | W | | | | | | | | |
| 0x000D PITLD1 (Low) | R | PLD7 | PLD6 | PLD5 | PLD4 | PLD3 | PLD2 | PLD1 | PLD0 |
| | W | | | | | | | | |
| 0x000E PITCNT1 (High) | R | PCNT15 | PCNT14 | PCNT13 | PCNT12 | PCNT11 | PCNT10 | PCNT9 | PCNT8 |
| | W | | | | | | | | |

= Unimplemented or Reserved

**Figure 18-2. PIT Register Summary (Sheet 1 of 2)**

| Register Name | | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|---|
| 0x000F PITCNT1 (Low) | R | PCNT7 | PCNT6 | PCNT5 | PCNT4 | PCNT3 | PCNT2 | PCNT1 | PCNT0 |
| | W | | | | | | | | |
| 0x0010 PITLD2 (High) | R | PLD15 | PLD14 | PLD13 | PLD12 | PLD11 | PLD10 | PLD9 | PLD8 |
| | W | | | | | | | | |
| 0x0011 PITLD2 (Low) | R | PLD7 | PLD6 | PLD5 | PLD4 | PLD3 | PLD2 | PLD1 | PLD0 |
| | W | | | | | | | | |
| 0x0012 PITCNT2 (High) | R | PCNT15 | PCNT14 | PCNT13 | PCNT12 | PCNT11 | PCNT10 | PCNT9 | PCNT8 |
| | W | | | | | | | | |
| 0x0013 PITCNT2 (Low) | R | PCNT7 | PCNT6 | PCNT5 | PCNT4 | PCNT3 | PCNT2 | PCNT1 | PCNT0 |
| | W | | | | | | | | |
| 0x0014 PITLD3 (High) | R | PLD15 | PLD14 | PLD13 | PLD12 | PLD11 | PLD10 | PLD9 | PLD8 |
| | W | | | | | | | | |
| 0x0015 PITLD3 (Low) | R | PLD7 | PLD6 | PLD5 | PLD4 | PLD3 | PLD2 | PLD1 | PLD0 |
| | W | | | | | | | | |
| 0x0016 PITCNT3 (High) | R | PCNT15 | PCNT14 | PCNT13 | PCNT12 | PCNT11 | PCNT10 | PCNT9 | PCNT8 |
| | W | | | | | | | | |
| 0x0017 PITCNT3 (Low) | R | PCNT7 | PCNT6 | PCNT5 | PCNT4 | PCNT3 | PCNT2 | PCNT1 | PCNT0 |
| | W | | | | | | | | |
| 0x0018–0x0027 RESERVED | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | W | | | | | | | | |

= Unimplemented or Reserved

**Figure 18-2. PIT Register Summary (Sheet 2 of 2)**

### 18.3.0.1 PIT Control and Force Load Micro Timer Register (PITCFLMT)

Module Base + 0x0000

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | PITE | PITSWAI | PITFRZ | 0 | 0 | 0 | 0 | 0 |
| W | | | | | | | PFLMT1 | PFLMT0 |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented or Reserved

**Figure 18-3. PIT Control and Force Load Micro Timer Register (PITCFLMT)**

Read: Anytime

Write: Anytime; writes to the reserved bits have no effect

**Table 18-2. PITCFLMT Field Descriptions**

| Field | Description |
|---|---|
| 7<br>PITE | **PIT Module Enable Bit** — This bit enables the PIT module. If PITE is cleared, the PIT module is disabled and flag bits in the PITTF register are cleared. When PITE is set, individually enabled timers (PCE set) start down-counting with the corresponding load register values.<br>0 PIT disabled (lower power consumption).<br>1 PIT is enabled. |
| 6<br>PITSWAI | **PIT Stop in Wait Mode Bit** — This bit is used for power conservation while in wait mode.<br>0 PIT operates normally in wait mode<br>1 PIT clock generation stops and freezes the PIT module when in wait mode |
| 5<br>PITFRZ | **PIT Counter Freeze while in Freeze Mode Bit** — When during debugging a breakpoint (freeze mode) is encountered it is useful in many cases to freeze the PIT counters to avoid e.g. interrupt generation. The PITFRZ bit controls the PIT operation while in freeze mode.<br>0 PIT operates normally in freeze mode<br>1 PIT counters are stalled when in freeze mode |
| 1:0<br>PFLMT[1:0] | **PIT Force Load Bits for Micro Timer 1:0** — These bits have only an effect if the corresponding micro timer is active and if the PIT module is enabled (PITE set). Writing a one into a PFLMT bit loads the corresponding 8-bit micro timer load register into the 8-bit micro timer down-counter. Writing a zero has no effect. Reading these bits will always return zero.<br>**Note:** A micro timer force load affects all timer channels that use the corresponding micro time base. |

## 18.3.0.2 PIT Force Load Timer Register (PITFLT)

Module Base + 0x0001

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| W | | | | | PFLT3 | PFLT2 | PFLT1 | PFLT0 |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 18-4. PIT Force Load Timer Register (PITFLT)**

Read: Anytime

Write: Anytime

**Table 18-3. PITFLT Field Descriptions**

| Field | Description |
|---|---|
| 3:0<br>PFLT[3:0] | **PIT Force Load Bits for Timer 3-0** — These bits have only an effect if the corresponding timer channel (PCE set) is enabled and if the PIT module is enabled (PITE set). Writing a one into a PFLT bit loads the corresponding 16-bit timer load register into the 16-bit timer down-counter. Writing a zero has no effect. Reading these bits will always return zero. |

### 18.3.0.3 PIT Channel Enable Register (PITCE)

Module Base + 0x0002

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | 0 | 0 | PCE3 | PCE2 | PCE1 | PCE0 |
| W | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 18-5. PIT Channel Enable Register (PITCE)**

Read: Anytime

Write: Anytime

**Table 18-4. PITCE Field Descriptions**

| Field | Description |
|---|---|
| 3:0<br>PCE[3:0] | **PIT Enable Bits for Timer Channel 3:0** — These bits enable the PIT channels 3-0. If PCE is cleared, the PIT channel is disabled and the corresponding flag bit in the PITTF register is cleared. When PCE is set, and if the PIT module is enabled (PITE = 1) the 16-bit timer counter is loaded with the start count value and starts down-counting.<br>0 The corresponding PIT channel is disabled.<br>1 The corresponding PIT channel is enabled. |

### 18.3.0.4 PIT Multiplex Register (PITMUX)

Module Base + 0x0003

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | 0 | 0 | PMUX3 | PMUX2 | PMUX1 | PMUX0 |
| W | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 18-6. PIT Multiplex Register (PITMUX)**

Read: Anytime

Write: Anytime

**Table 18-5. PITMUX Field Descriptions**

| Field | Description |
|---|---|
| 3:0<br>PMUX[3:0] | **PIT Multiplex Bits for Timer Channel 3:0** — These bits select if the corresponding 16-bit timer is connected to micro time base 1 or 0. If PMUX is modified, the corresponding 16-bit timer is switched to the other micro time base immediately.<br>0 The corresponding 16-bit timer counts with micro time base 0.<br>1 The corresponding 16-bit timer counts with micro time base 1. |

### 18.3.0.5 PIT Interrupt Enable Register (PITINTE)

Module Base + 0x0004

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | 0 | 0 | PINTE3 | PINTE2 | PINTE1 | PINTE0 |
| W | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 18-7. PIT Interrupt Enable Register (PITINTE)**

Read: Anytime

Write: Anytime

**Table 18-6. PITINTE Field Descriptions**

| Field | Description |
|---|---|
| 3:0<br>PINTE[3:0] | **PIT Time-out Interrupt Enable Bits for Timer Channel 3:0** — These bits enable an interrupt service request whenever the time-out flag PTF of the corresponding PIT channel is set. When an interrupt is pending (PTF set) enabling the interrupt will immediately cause an interrupt. To avoid this, the corresponding PTF flag has to be cleared first.<br>0 Interrupt of the corresponding PIT channel is disabled.<br>1 Interrupt of the corresponding PIT channel is enabled. |

### 18.3.0.6 PIT Time-Out Flag Register (PITTF)

Module Base + 0x0005

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | 0 | 0 | PTF3 | PTF2 | PTF1 | PTF0 |
| W | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 18-8. PIT Time-Out Flag Register (PITTF)**

Read: Anytime

Write: Anytime (write to clear)

**Table 18-7. PITTF Field Descriptions**

| Field | Description |
|---|---|
| 3:0<br>PTF[3:0] | **PIT Time-out Flag Bits for Timer Channel 3:0** — PTF is set when the corresponding 16-bit timer modulus down-counter and the selected 8-bit micro timer modulus down-counter have counted to zero. The flag can be cleared by writing a one to the flag bit. Writing a zero has no effect. If flag clearing by writing a one and flag setting happen in the same bus clock cycle, the flag remains set. The flag bits are cleared if the PIT module is disabled or if the corresponding timer channel is disabled.<br>0 Time-out of the corresponding PIT channel has not yet occurred.<br>1 Time-out of the corresponding PIT channel has occurred. |

### 18.3.0.7 PIT Micro Timer Load Register 0 to 1 (PITMTLD0–1)

Module Base + 0x0006

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | PMTLD7 | PMTLD6 | PMTLD5 | PMTLD4 | PMTLD3 | PMTLD2 | PMTLD1 | PMTLD0 |
| W | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 18-9. PIT Micro Timer Load Register 0 (PITMTLD0)**

Module Base + 0x0007

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | PMTLD7 | PMTLD6 | PMTLD5 | PMTLD4 | PMTLD3 | PMTLD2 | PMTLD1 | PMTLD0 |
| W | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 18-10. PIT Micro Timer Load Register 1 (PITMTLD1)**

Read: Anytime

Write: Anytime

**Table 18-8. PITMTLD0–1 Field Descriptions**

| Field | Description |
|---|---|
| 7:0<br>PMTLD[7:0] | **PIT Micro Timer Load Bits 7:0** — These bits set the 8-bit modulus down-counter load value of the micro timers. Writing a new value into the PITMTLD register will not restart the timer. When the micro timer has counted down to zero, the PMTLD register value will be loaded. The PFLMT bits in the PITCFLMT register can be used to immediately update the count register with the new value if an immediate load is desired. |

### 18.3.0.8 PIT Load Register 0 to 3 (PITLD0–3)

Module Base + 0x0008, 0x0009

| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | PLD15 | PLD14 | PLD13 | PLD12 | PLD11 | PLD10 | PLD9 | PLD8 | PLD7 | PLD6 | PLD5 | PLD4 | PLD3 | PLD2 | PLD1 | PLD0 |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 18-11. PIT Load Register 0 (PITLD0)**

Module Base + 0x000C, 0x000D

| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | PLD15 | PLD14 | PLD13 | PLD12 | PLD11 | PLD10 | PLD9 | PLD8 | PLD7 | PLD6 | PLD5 | PLD4 | PLD3 | PLD2 | PLD1 | PLD0 |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 18-12. PIT Load Register 1 (PITLD1)**

Module Base + 0x0010, 0x0011

| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | PLD15 | PLD14 | PLD13 | PLD12 | PLD11 | PLD10 | PLD9 | PLD8 | PLD7 | PLD6 | PLD5 | PLD4 | PLD3 | PLD2 | PLD1 | PLD0 |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 18-13. PIT Load Register 2 (PITLD2)**

Module Base + 0x0014, 0x0015

| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | PLD15 | PLD14 | PLD13 | PLD12 | PLD11 | PLD10 | PLD9 | PLD8 | PLD7 | PLD6 | PLD5 | PLD4 | PLD3 | PLD2 | PLD1 | PLD0 |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 18-14. PIT Load Register 3 (PITLD3)**

Read: Anytime

Write: Anytime

**Table 18-9. PITLD0–3 Field Descriptions**

| Field | Description |
|---|---|
| 15:0<br>PLD[15:0] | **PIT Load Bits 15:0** — These bits set the 16-bit modulus down-counter load value. Writing a new value into the PITLD register must be a 16-bit access, to ensure data consistency. It will not restart the timer. When the timer has counted down to zero the PTF time-out flag will be set and the register value will be loaded. The PFLT bits in the PITFLT register can be used to immediately update the count register with the new value if an immediate load is desired. |

## 18.3.0.9 PIT Count Register 0 to 3 (PITCNT0–3)

Module Base + 0x000A, 0x000B

| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | PCNT 15 | PCNT 14 | PCNT 13 | PCNT 12 | PCNT 11 | PCNT 10 | PCN T9 | PCN T8 | PCN T7 | PCN T6 | PCN T5 | PCN T4 | PCN T3 | PCN T2 | PCN T1 | PCN T0 |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 18-15. PIT Count Register 0 (PITCNT0)**

Module Base + 0x000E, 0x000F

| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | PCNT 15 | PCNT 14 | PCNT 13 | PCNT 12 | PCNT 11 | PCNT 10 | PCN T9 | PCN T8 | PCN T7 | PCN T6 | PCN T5 | PCN T4 | PCN T3 | PCN T2 | PCN T1 | PCN T0 |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 18-16. PIT Count Register 1 (PITCNT1)**

Module Base + 0x0012, 0x0013

| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | PCNT 15 | PCNT 14 | PCNT 13 | PCNT 12 | PCNT 11 | PCNT 10 | PCN T9 | PCN T8 | PCN T7 | PCN T6 | PCN T5 | PCN T4 | PCN T3 | PCN T2 | PCN T1 | PCN T0 |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 18-17. PIT Count Register 2 (PITCNT2)**

Module Base + 0x0016, 0x0017

| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | PCNT 15 | PCNT 14 | PCNT 13 | PCNT 12 | PCNT 11 | PCNT 10 | PCN T9 | PCN T8 | PCN T7 | PCN T6 | PCN T5 | PCN T4 | PCN T3 | PCN T2 | PCN T1 | PCN T0 |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 18-18. PIT Count Register 3 (PITCNT3)**

Read: Anytime

Write: Has no meaning or effect

**Table 18-10. PITCNT0–3 Field Descriptions**

| Field | Description |
|---|---|
| 15:0<br>PCNT[15:0] | **PIT Count Bits 15-0** — These bits represent the current 16-bit modulus down-counter value. The read access for the count register must take place in one clock cycle as a 16-bit access. |

## 18.4 Functional Description

Figure 18-19 shows a detailed block diagram of the PIT module. The main parts of the PIT are status, control and data registers, two 8-bit down-counters, four 16-bit down-counters and an interrupt/trigger interface.

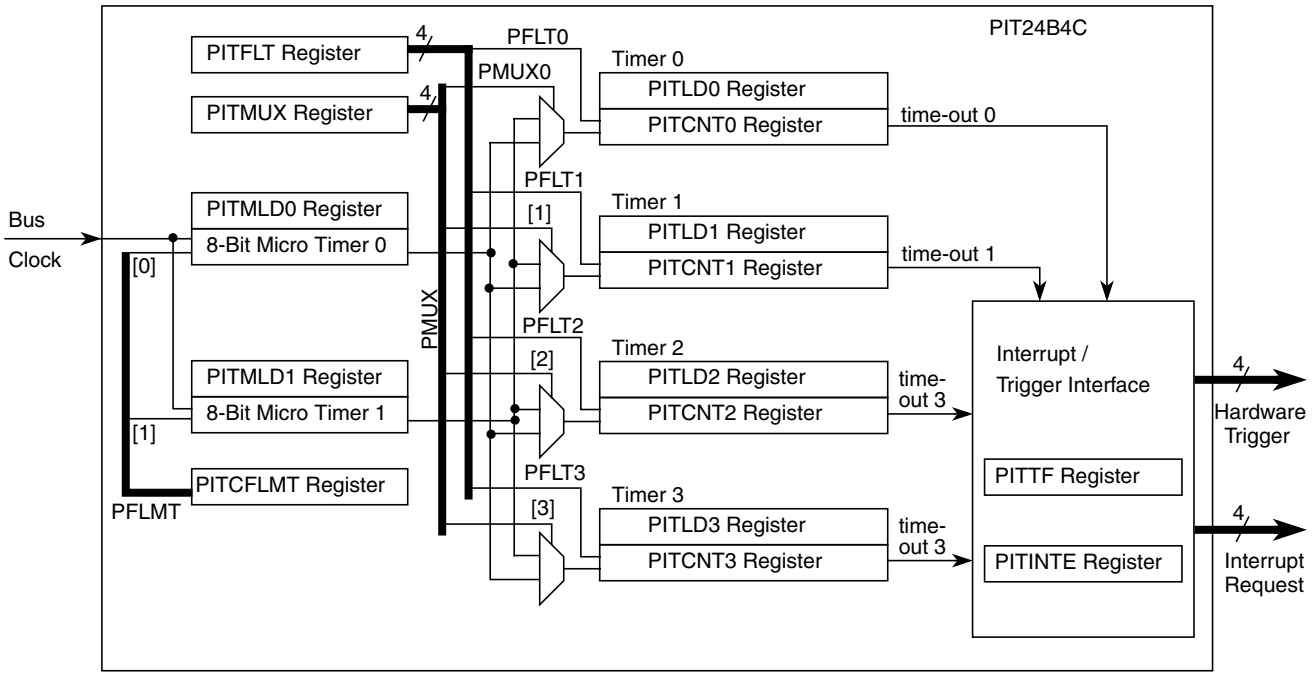



**Figure 18-19. PIT24B4C Detailed Block Diagram**

### 18.4.1 Timer

As shown in Figure 18-1 and Figure 18-19, the 24-bit timers are built in a two-stage architecture with four 16-bit modulus down-counters and two 8-bit modulus down-counters. The 16-bit timers are clocked with two selectable micro time bases which are generated with 8-bit modulus down-counters. Each 16-bit timer is connected to micro time base 0 or 1 via the PMUX[3:0] bit setting in the PIT Multiplex (PITMUX) register.

A timer channel is enabled if the module enable bit PITE in the PIT control and force load micro timer (PITCFLMT) register is set and if the corresponding PCE bit in the PIT channel enable (PITCE) register is set. Two 8-bit modulus down-counters are used to generate two micro time bases. As soon as a micro time base is selected for an enabled timer channel, the corresponding micro timer modulus down-counter will load its start value as specified in the PITMTLD0 or PITMTLD1 register and will start down-counting.

Whenever the micro timer down-counter has counted to zero the PITMTLD register is reloaded and the connected 16-bit modulus down-counters count one cycle.

Whenever a 16-bit timer counter and the connected 8-bit micro timer counter have counted to zero, the PITLD register is reloaded and the corresponding time-out flag PTF in the PIT time-out flag (PITTF) register is set, as shown in Figure 18-20. The time-out period is a function of the timer load (PITLD) and micro timer load (PITMTLD) registers and the bus clock $f_{BUS}$:

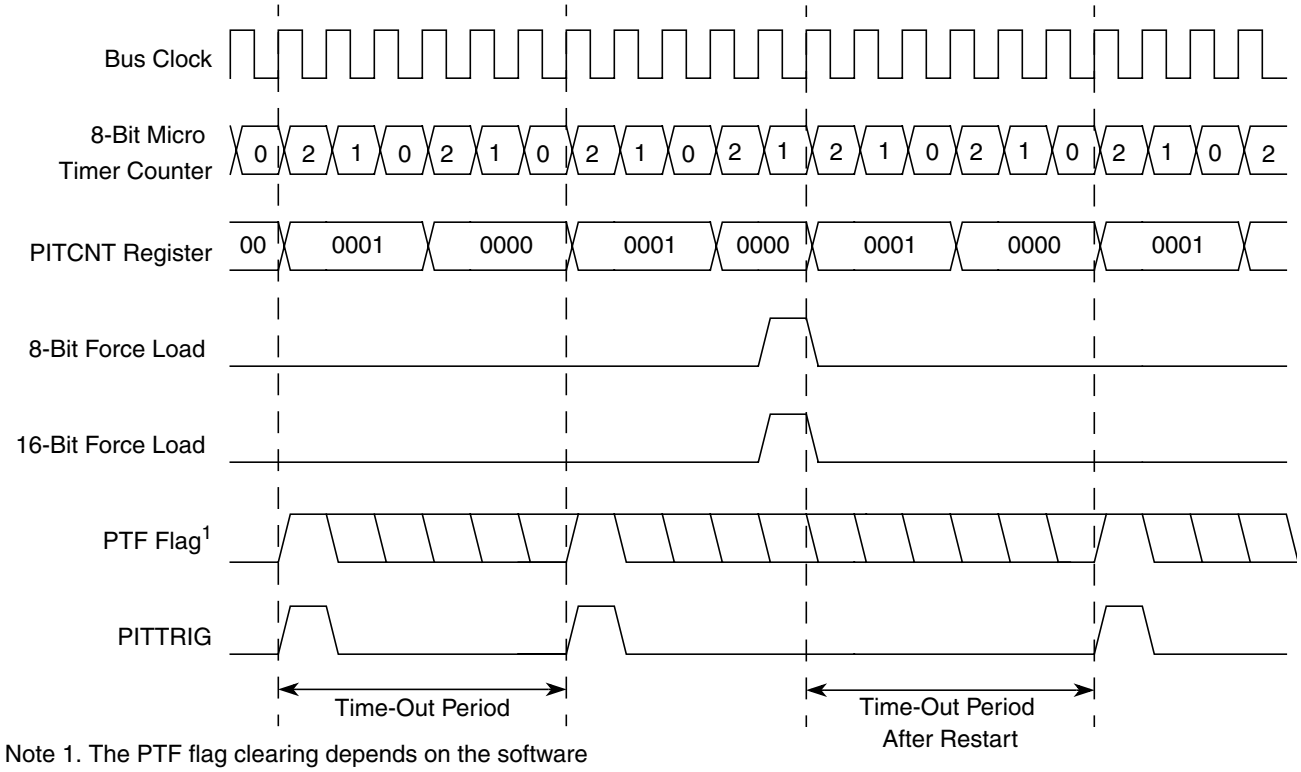
time-out period = (PITMTLD + 1) * (PITLD + 1) / $f_{BUS}$.

For example, for a 40 MHz bus clock, the maximum time-out period equals:

256 * 65536 * 25 ns = 419.43 ms.

The current 16-bit modulus down-counter value can be read via the PITCNT register. The micro timer down-counter values cannot be read.

The 8-bit micro timers can individually be restarted by writing a one to the corresponding force load micro timer PFLMT bits in the PIT control and force load micro timer (PITCFLMT) register. The 16-bit timers can individually be restarted by writing a one to the corresponding force load timer PFLT bits in the PIT forceload timer (PITFLT) register. If desired, any group of timers and micro timers can be restarted at the same time by using one 16-bit write to the adjacent PITCFLMT and PITFLT registers with the relevant bits set, as shown in Figure 18-20.




Note 1. The PTF flag clearing depends on the software

**Figure 18-20. PIT Trigger and Flag Signal Timing**

### 18.4.2 Interrupt Interface

Each time-out event can be used to trigger an interrupt service request. For each timer channel, an individual bit PINTE in the PIT interrupt enable (PITINTE) register exists to enable this feature. If PINTE is set, an interrupt service is requested whenever the corresponding time-out flag PTF in the PIT time-out flag (PITTF) register is set. The flag can be cleared by writing a one to the flag bit.

**NOTE**

Be careful when resetting the PITE, PINTE or PITCE bits in case of pending PIT interrupt requests, to avoid spurious interrupt requests.

### 18.4.3 Hardware Trigger

The PIT module contains four hardware trigger signal lines PITTRIG[3:0], one for each timer channel. These signals can be connected on SoC level to peripheral modules enabling e.g. periodic ATD conversion (please refer to the device overview for the mapping of PITTRIG[3:0] signals to peripheral modules).

Whenever a timer channel time-out is reached, the corresponding PTF flag is set and the corresponding trigger signal PITTRIG triggers a rising edge. The trigger feature requires a minimum time-out period of two bus clock cycles because the trigger is asserted high for at least one bus clock cycle. For load register values PITLD = 0x0001 and PITMTLD = 0x0002 the flag setting, trigger timing and a restart with force load is shown in Figure 18-20.

## 18.5 Initialization

### 18.5.1 Startup

Set the configuration registers before the PITE bit in the PITCFLMT register is set. Before PITE is set, the configuration registers can be written in arbitrary order.

### 18.5.2 Shutdown

When the PITCE register bits, the PITINTE register bits or the PITE bit in the PITCFLMT register are cleared, the corresponding PIT interrupt flags are cleared. In case of a pending PIT interrupt request, a spurious interrupt can be generated. Two strategies, which avoid spurious interrupts, are recommended:

1. Reset the PIT interrupt flags only in an ISR. When entering the ISR, the I mask bit in the CCR is set automatically. The I mask bit must not be cleared before the PIT interrupt flags are cleared.
2. After setting the I mask bit with the SEI instruction, the PIT interrupt flags can be cleared. Then clear the I mask bit with the CLI instruction to re-enable interrupts.

### 18.5.3 Flag Clearing

A flag is cleared by writing a one to the flag bit. Always use store or move instructions to write a one in certain bit positions. Do not use the BSET instructions. Do not use any C-constructs that compile to BSET instructions. “BSET flag_register, #mask” must not be used for flag clearing because BSET is a read-

modify-write instruction which writes back the “bit-wise or” of the flag_register and the mask into the flag_register. BSET would clear all flag bits that were set, independent from the mask.

For example, to clear flag bit 0 use: MOVB #$01,PITTF.

## 18.6 Application Information

To get started quickly with the PIT24B4C module this section provides a small code example how to use the block. Please note that the example provided is only one specific case out of the possible configurations and implementations.

Functionality: Generate an PIT interrupt on channel 0 every 500 PIT clock cycles.

```
                ORG     CODESTART               ; place the program into specific
                                                ; range (to be selected)
                LDS     RAMEND                  ; load stack pointer to top of RAM
                MOVW    #CH0_ISR,VEC_PIT_CH0    ; Change value of channel 0 ISR adr

; ******************** Start PIT Initialization ***************************************************

                CLR     PITCFLMT                ; disable PIT
                MOVB    #$01,PITCE              ; enable timer channel 0
                CLR     PITMUX                  ; ch0 connected to micro timer 0
                MOVB    #$63,PITMTLD0           ; micro time base 0 equals 100 clock cycles
                MOVW    #$0004,PITLD0           ; time base 0 eq. 5 micro time bases 0 =5*100 = 500
                MOVB    #$01,PITINTE            ; enable interupt channel 0
                MOVB    #$80,PITCFLMT           ; enable PIT
                CLI                             ; clear Interupt disable Mask bit

;******************** Main Program ***************************************************************

MAIN:           BRA *                           ; loop until interrupt

;******************** Channel 0 Interupt Routine *************************************************

CH0_ISR:        LDAA    PITTF                   ; 8 bit read of PIT time out flags
                MOVB    #$01,PITTF              ; clear PIT channel 0 time out flag
                RTI                             ; return to MAIN
```

# Chapter 19
# Pulse-Width Modulator (S12PWM8B8CV1)

**Table 19-1. Revision History**

| Revision Number | Revision Date | Sections Affected | Description of Changes |
|---|---|---|---|
| 1.1 | 28 Sepr 2012 | 1.4.2.6 | deleted the blank in the doc |

## 19.1 Introduction

The PWM definition is based on the HC12 PWM definitions. It contains the basic features from the HC11 with some of the enhancements incorporated on the HC12: center aligned output mode and four available clock sources.The PWM module has eight channels with independent control of left and center aligned outputs on each channel.

Each of the eight channels has a programmable period and duty cycle as well as a dedicated counter. A flexible clock select scheme allows a total of four different clock sources to be used with the counters. Each of the modulators can create independent continuous waveforms with software-selectable duty rates from 0% to 100%. The PWM outputs can be programmed as left aligned outputs or center aligned outputs.

### 19.1.1 Features

The PWM block includes these distinctive features:

- Eight independent PWM channels with programmable period and duty cycle
- Dedicated counter for each PWM channel
- Programmable PWM enable/disable for each channel
- Software selection of PWM duty pulse polarity for each channel
- Period and duty cycle are double buffered. Change takes effect when the end of the effective period is reached (PWM counter reaches zero) or when the channel is disabled.
- Programmable center or left aligned outputs on individual channels
- Eight 8-bit channel or four 16-bit channel PWM resolution
- Four clock sources (A, B, SA, and SB) provide for a wide range of frequencies
- Programmable clock select logic
- Emergency shutdown

### 19.1.2 Modes of Operation

There is a software programmable option for low power consumption in wait mode that disables the input clock to the prescaler.

In freeze mode there is a software programmable option to disable the input clock to the prescaler. This is useful for emulation.

### 19.1.3 Block Diagram

Figure 19-1 shows the block diagram for the 8-bit 8-channel PWM block.

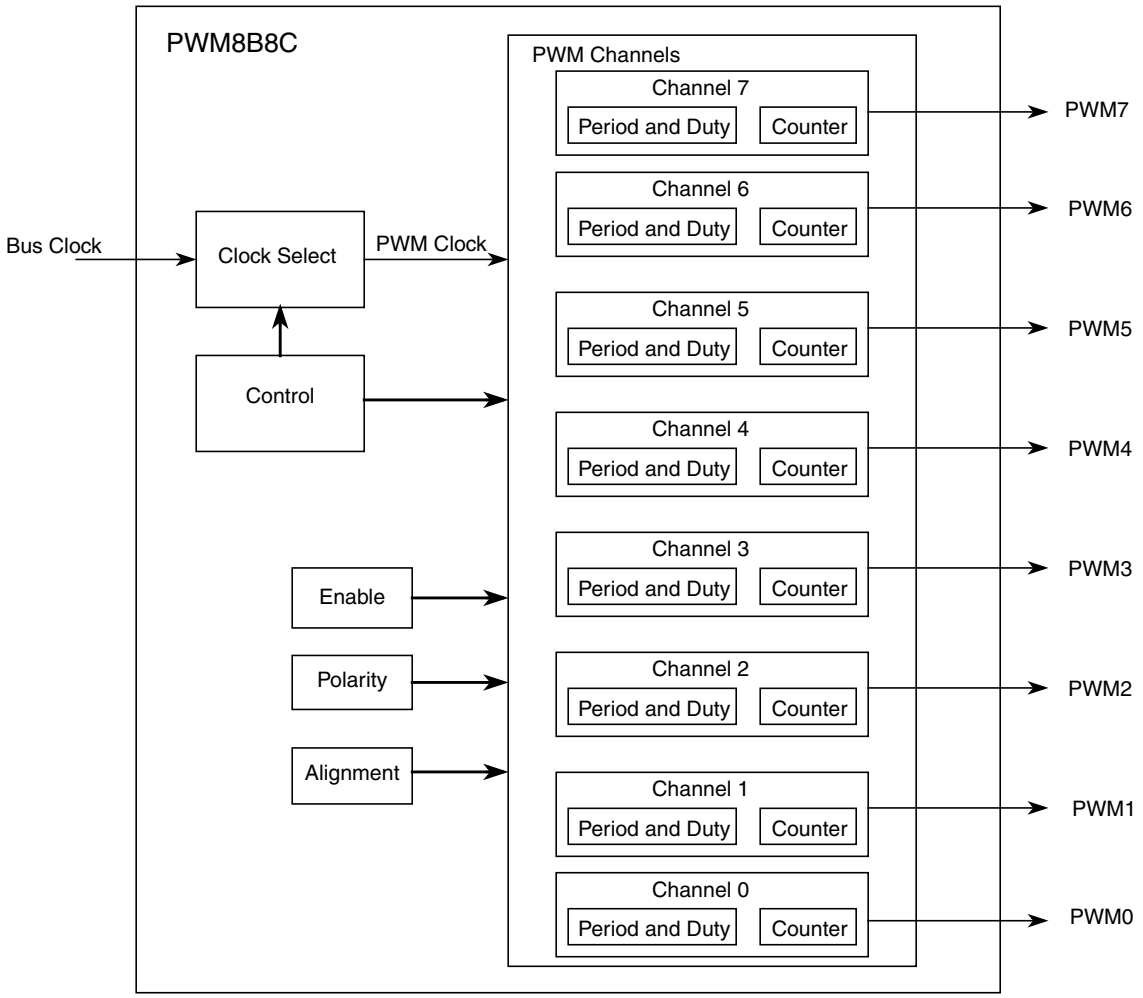



**Figure 19-1. PWM Block Diagram**

## 19.2 External Signal Description

The PWM module has a total of 8 external pins.

### 19.2.1 PWM7 — PWM Channel 7

This pin serves as waveform output of PWM channel 7 and as an input for the emergency shutdown feature.

### 19.2.2 PWM6 — PWM Channel 6

This pin serves as waveform output of PWM channel 6.

### 19.2.3 PWM5 — PWM Channel 5

This pin serves as waveform output of PWM channel 5.

### 19.2.4 PWM4 — PWM Channel 4

This pin serves as waveform output of PWM channel 4.

### 19.2.5 PWM3 — PWM Channel 3

This pin serves as waveform output of PWM channel 3.

### 19.2.6 PWM3 — PWM Channel 2

This pin serves as waveform output of PWM channel 2.

### 19.2.7 PWM3 — PWM Channel 1

This pin serves as waveform output of PWM channel 1.

### 19.2.8 PWM3 — PWM Channel 0

This pin serves as waveform output of PWM channel 0.

## 19.3 Memory Map and Register Definition

This section describes in detail all the registers and register bits in the PWM module.

The special-purpose registers and register bit functions that are not normally available to device end users, such as factory test control registers and reserved registers, are clearly identified by means of shading the appropriate portions of address maps and register diagrams. Notes explaining the reasons for restricting access to the registers and functions are also explained in the individual register descriptions.

### 19.3.1 Module Memory Map

This section describes the content of the registers in the PWM module. The base address of the PWM module is determined at the MCU level when the MCU is defined. The register decode map is fixed and begins at the first address of the module address offset. The figure below shows the registers associated with the PWM and their relative offset from the base address. The register detail description follows the order they appear in the register map.

Reserved bits within a register will always read as 0 and the write will be unimplemented. Unimplemented functions are indicated by shading the bit. .

**NOTE**

Register Address = Base Address + Address Offset, where the Base Address is defined at the MCU level and the Address Offset is defined at the module level.

## 19.3.2 Register Descriptions

This section describes in detail all the registers and register bits in the PWM module.

| Register Name | | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|---|
| 0x0000 PWME | R | PWME7 | PWME6 | PWME5 | PWME4 | PWME3 | PWME2 | PWME1 | PWME0 |
| | W | | | | | | | | |
| 0x0001 PWMPOL | R | PPOL7 | PPOL6 | PPOL5 | PPOL4 | PPOL3 | PPOL2 | PPOL1 | PPOL0 |
| | W | | | | | | | | |
| 0x0002 PWMCLK | R | PCLK7 | PCLKL6 | PCLK5 | PCLK4 | PCLK3 | PCLK2 | PCLK1 | PCLK0 |
| | W | | | | | | | | |
| 0x0003 PWMPRCLK | R | 0 | PCKB2 | PCKB1 | PCKB0 | 0 | PCKA2 | PCKA1 | PCKA0 |
| | W | | | | | | | | |
| 0x0004 PWMCAE | R | CAE7 | CAE6 | CAE5 | CAE4 | CAE3 | CAE2 | CAE1 | CAE0 |
| | W | | | | | | | | |
| 0x0005 PWMCTL | R | CON67 | CON45 | CON23 | CON01 | PSWAI | PFRZ | 0 | 0 |
| | W | | | | | | | | |
| 0x0006 PWMTST(1) | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | W | | | | | | | | |
| 0x0007 PWMPRSC1 | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | W | | | | | | | | |
| 0x0008 PWMSCLA | R | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
| | W | | | | | | | | |
| 0x0009 PWMSCLB | R | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
| | W | | | | | | | | |
| 0x000A PWMSCNTA 1 | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | W | | | | | | | | |

= Unimplemented or Reserved

**Figure 19-2. PWM Register Summary (Sheet 1 of 3)**

| Register Name | | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|---|
| 0x000B PWMSCNTB [1] | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | W | | | | | | | | |
| 0x000C PWMCNT0 | R | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
| | W | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0x000D PWMCNT1 | R | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
| | W | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0x000E PWMCNT2 | R | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
| | W | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0x000F PWMCNT3 | R | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
| | W | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0x0010 PWMCNT4 | R | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
| | W | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0x0011 PWMCNT5 | R | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
| | W | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0x0012 PWMCNT6 | R | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
| | W | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0x0013 PWMCNT7 | R | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
| | W | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0x0014 PWMPER0 | R | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
| | W | | | | | | | | |
| 0x0015 PWMPER1 | R | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
| | W | | | | | | | | |
| 0x0016 PWMPER2 | R | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
| | W | | | | | | | | |
| 0x0017 PWMPER3 | R | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
| | W | | | | | | | | |
| 0x0018 PWMPER4 | R | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
| | W | | | | | | | | |
| 0x0019 PWMPER5 | R | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
| | W | | | | | | | | |

= Unimplemented or Reserved

**Figure 19-2. PWM Register Summary (Sheet 2 of 3)**

| Register Name | | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|---|
| 0x001A PWMPER6 | R<br>W | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
| 0x001B PWMPER7 | R<br>W | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
| 0x001C PWMDTY0 | R<br>W | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
| 0x001D PWMDTY1 | R<br>W | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
| 0x001E PWMDTY2 | R<br>W | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
| 0x001F PWMDTY3 | R<br>W | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
| 0x0020 PWMDTY4 | R<br>W | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
| 0x0021 PWMDTY5 | R<br>W | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
| 0x0022 PWMDTY6 | R<br>W | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
| 0x0023 PWMDTY7 | R<br>W | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
| 0x0024 PWMSDN | R<br>W | PWMIF | PWMIE | 0<br>PWMRSTRT | PWMLVL | 0 | PWM7IN | PWM7INL | PWM7ENA |

= Unimplemented or Reserved

**Figure 19-2. PWM Register Summary (Sheet 3 of 3)**

1. Intended for factory test purposes only.

### 19.3.2.1 PWM Enable Register (PWME)

Each PWM channel has an enable bit (PWMEx) to start its waveform output. When any of the PWMEx bits are set (PWMEx = 1), the associated PWM output is enabled immediately. However, the actual PWM waveform is not available on the associated PWM output until its clock source begins its next cycle due to the synchronization of PWMEx and the clock source.

**NOTE**

The first PWM cycle after enabling the channel can be irregular.

An exception to this is when channels are concatenated. Once concatenated mode is enabled (CONxx bits set in PWMCTL register), enabling/disabling the corresponding 16-bit PWM channel is controlled by the low order PWMEx bit.In this case, the high order bytes PWMEx bits have no effect and their corresponding PWM output lines are disabled.

While in run mode, if all eight PWM channels are disabled (PWME7–0 = 0), the prescaler counter shuts off for power savings.

Module Base + 0x0000

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | PWME7 | PWME6 | PWME5 | PWME4 | PWME3 | PWME2 | PWME1 | PWME0 |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 19-3. PWM Enable Register (PWME)**

Read: Anytime

Write: Anytime

**Table 19-2. PWME Field Descriptions**

| Field | Description |
|---|---|
| 7<br>PWME7 | **Pulse Width Channel 7 Enable**<br>0 Pulse width channel 7 is disabled.<br>1 Pulse width channel 7 is enabled. The pulse modulated signal becomes available at PWM output bit 7 when its clock source begins its next cycle. |
| 6<br>PWME6 | **Pulse Width Channel 6 Enable**<br>0 Pulse width channel 6 is disabled.<br>1 Pulse width channel 6 is enabled. The pulse modulated signal becomes available at PWM output bit6 when its clock source begins its next cycle. If CON67=1, then bit has no effect and PWM output line 6 is disabled. |
| 5<br>PWME5 | **Pulse Width Channel 5 Enable**<br>0 Pulse width channel 5 is disabled.<br>1 Pulse width channel 5 is enabled. The pulse modulated signal becomes available at PWM output bit 5 when its clock source begins its next cycle. |
| 4<br>PWME4 | **Pulse Width Channel 4 Enable**<br>0 Pulse width channel 4 is disabled.<br>1 Pulse width channel 4 is enabled. The pulse modulated signal becomes available at PWM, output bit 4 when its clock source begins its next cycle. If CON45 = 1, then bit has no effect and PWM output bit4 is disabled. |
| 3<br>PWME3 | **Pulse Width Channel 3 Enable**<br>0 Pulse width channel 3 is disabled.<br>1 Pulse width channel 3 is enabled. The pulse modulated signal becomes available at PWM, output bit 3 when its clock source begins its next cycle. |
| 2<br>PWME2 | **Pulse Width Channel 2 Enable**<br>0 Pulse width channel 2 is disabled.<br>1 Pulse width channel 2 is enabled. The pulse modulated signal becomes available at PWM, output bit 2 when its clock source begins its next cycle. If CON23 = 1, then bit has no effect and PWM output bit2 is disabled. |

**Table 19-2. PWME Field Descriptions (continued)**

| Field | Description |
|---|---|
| 1<br>PWME1 | **Pulse Width Channel 1 Enable**<br>0 Pulse width channel 1 is disabled.<br>1 Pulse width channel 1 is enabled. The pulse modulated signal becomes available at PWM, output bit 1 when its clock source begins its next cycle. |
| 0<br>PWME0 | **Pulse Width Channel 0 Enable**<br>0 Pulse width channel 0 is disabled.<br>1 Pulse width channel 0 is enabled. The pulse modulated signal becomes available at PWM, output bit 0 when its clock source begins its next cycle. If CON01 = 1, then bit has no effect and PWM output line0 is disabled. |

### 19.3.2.2 PWM Polarity Register (PWMPOL)

The starting polarity of each PWM channel waveform is determined by the associated PPOLx bit in the PWMPOL register. If the polarity bit is one, the PWM channel output is high at the beginning of the cycle and then goes low when the duty count is reached. Conversely, if the polarity bit is zero, the output starts low and then goes high when the duty count is reached.

Module Base + 0x0001

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | PPOL7 | PPOL6 | PPOL5 | PPOL4 | PPOL3 | PPOL2 | PPOL1 | PPOL0 |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 19-4. PWM Polarity Register (PWMPOL)**

Read: Anytime

Write: Anytime

**NOTE**

PPOLx register bits can be written anytime. If the polarity is changed while a PWM signal is being generated, a truncated or stretched pulse can occur during the transition

**Table 19-3. PWMPOL Field Descriptions**

| Field | Description |
|---|---|
| 7–0<br>PPOL[7:0] | **Pulse Width Channel 7–0 Polarity Bits**<br>0 PWM channel 7–0 outputs are low at the beginning of the period, then go high when the duty count is reached.<br>1 PWM channel 7–0 outputs are high at the beginning of the period, then go low when the duty count is reached. |

### 19.3.2.3 PWM Clock Select Register (PWMCLK)

Each PWM channel has a choice of two clocks to use as the clock source for that channel as described below.

Module Base + 0x0002

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | PCLK7 | PCLKL6 | PCLK5 | PCLK4 | PCLK3 | PCLK2 | PCLK1 | PCLK0 |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 19-5. PWM Clock Select Register (PWMCLK)**

Read: Anytime

Write: Anytime

**NOTE**

Register bits PCLK0 to PCLK7 can be written anytime. If a clock select is changed while a PWM signal is being generated, a truncated or stretched pulse can occur during the transition.

**Table 19-4. PWMCLK Field Descriptions**

| Field | Description |
|---|---|
| 7<br>PCLK7 | **Pulse Width Channel 7 Clock Select**<br>0 Clock B is the clock source for PWM channel 7.<br>1 Clock SB is the clock source for PWM channel 7. |
| 6<br>PCLK6 | **Pulse Width Channel 6 Clock Select**<br>0 Clock B is the clock source for PWM channel 6.<br>1 Clock SB is the clock source for PWM channel 6. |
| 5<br>PCLK5 | **Pulse Width Channel 5 Clock Select**<br>0 Clock A is the clock source for PWM channel 5.<br>1 Clock SA is the clock source for PWM channel 5. |
| 4<br>PCLK4 | **Pulse Width Channel 4 Clock Select**<br>0 Clock A is the clock source for PWM channel 4.<br>1 Clock SA is the clock source for PWM channel 4. |
| 3<br>PCLK3 | **Pulse Width Channel 3 Clock Select**<br>0 Clock B is the clock source for PWM channel 3.<br>1 Clock SB is the clock source for PWM channel 3. |
| 2<br>PCLK2 | **Pulse Width Channel 2 Clock Select**<br>0 Clock B is the clock source for PWM channel 2.<br>1 Clock SB is the clock source for PWM channel 2. |
| 1<br>PCLK1 | **Pulse Width Channel 1 Clock Select**<br>0 Clock A is the clock source for PWM channel 1.<br>1 Clock SA is the clock source for PWM channel 1. |
| 0<br>PCLK0 | **Pulse Width Channel 0 Clock Select**<br>0 Clock A is the clock source for PWM channel 0.<br>1 Clock SA is the clock source for PWM channel 0. |

### 19.3.2.4 PWM Prescale Clock Select Register (PWMPRCLK)

This register selects the prescale clock source for clocks A and B independently.

Module Base + 0x0003

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | 0 | PCKB2 | PCKB1 | PCKB0 | 0 | PCKA2 | PCKA1 | PCKA0 |
| W | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented or Reserved

**Figure 19-6. PWM Prescale Clock Select Register (PWMPRCLK)**

Read: Anytime

Write: Anytime

**NOTE**

PCKB2–0 and PCKA2–0 register bits can be written anytime. If the clock pre-scale is changed while a PWM signal is being generated, a truncated or stretched pulse can occur during the transition.

**Table 19-5. PWMPRCLK Field Descriptions**

| Field | Description |
|---|---|
| 6–4<br>PCKB[2:0] | **Prescaler Select for Clock B** — Clock B is one of two clock sources which can be used for channels 2, 3, 6, or 7. These three bits determine the rate of clock B, as shown in Table 19-6. |
| 2–0<br>PCKA[2:0] | **Prescaler Select for Clock A** — Clock A is one of two clock sources which can be used for channels 0, 1, 4 or 5. These three bits determine the rate of clock A, as shown in Table 19-7. |

**Table 19-6. Clock B Prescaler Selects**

| PCKB2 | PCKB1 | PCKB0 | Value of Clock B |
|---|---|---|---|
| 0 | 0 | 0 | Bus clock |
| 0 | 0 | 1 | Bus clock / 2 |
| 0 | 1 | 0 | Bus clock / 4 |
| 0 | 1 | 1 | Bus clock / 8 |
| 1 | 0 | 0 | Bus clock / 16 |
| 1 | 0 | 1 | Bus clock / 32 |
| 1 | 1 | 0 | Bus clock / 64 |
| 1 | 1 | 1 | Bus clock / 128 |

**Table 19-7. Clock A Prescaler Selects**

| PCKA2 | PCKA1 | PCKA0 | Value of Clock A |
|---|---|---|---|
| 0 | 0 | 0 | Bus clock |
| 0 | 0 | 1 | Bus clock / 2 |
| 0 | 1 | 0 | Bus clock / 4 |
| 0 | 1 | 1 | Bus clock / 8 |
| 1 | 0 | 0 | Bus clock / 16 |
| 1 | 0 | 1 | Bus clock / 32 |
| 1 | 1 | 0 | Bus clock / 64 |
| 1 | 1 | 1 | Bus clock / 128 |

### 19.3.2.5 PWM Center Align Enable Register (PWMCAE)

The PWMCAE register contains eight control bits for the selection of center aligned outputs or left aligned outputs for each PWM channel. If the CAEx bit is set to a one, the corresponding PWM output will be center aligned. If the CAEx bit is cleared, the corresponding PWM output will be left aligned. See Section 19.4.2.5, “Left Aligned Outputs” and Section 19.4.2.6, “Center Aligned Outputs” for a more detailed description of the PWM output modes.

Module Base + 0x0004

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | CAE7 | CAE6 | CAE5 | CAE4 | CAE3 | CAE2 | CAE1 | CAE0 |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 19-7. PWM Center Align Enable Register (PWMCAE)**

Read: Anytime

Write: Anytime

**NOTE**

Write these bits only when the corresponding channel is disabled.

**Table 19-8. PWMCAE Field Descriptions**

| Field | Description |
|---|---|
| 7–0<br>CAE[7:0] | **Center Aligned Output Modes on Channels 7–0**<br>0 Channels 7–0 operate in left aligned output mode.<br>1 Channels 7–0 operate in center aligned output mode. |

### 19.3.2.6 PWM Control Register (PWMCTL)

The PWMCTL register provides for various control of the PWM module.

Module Base + 0x0005

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | CON67 | CON45 | CON23 | CON01 | PSWAI | PFRZ | 0 | 0 |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented or Reserved

**Figure 19-8. PWM Control Register (PWMCTL)**

Read: Anytime

Write: Anytime

There are three control bits for concatenation, each of which is used to concatenate a pair of PWM channels into one 16-bit channel. When channels 6 and 7are concatenated, channel 6 registers become the high order bytes of the double byte channel. When channels 4 and 5 are concatenated, channel 4 registers become the high order bytes of the double byte channel. When channels 2 and 3 are concatenated, channel

2 registers become the high order bytes of the double byte channel. When channels 0 and 1 are concatenated, channel 0 registers become the high order bytes of the double byte channel.

See Section 19.4.2.7, "PWM 16-Bit Functions" for a more detailed description of the concatenation PWM Function.

**NOTE**

Change these bits only when both corresponding channels are disabled.

**Table 19-9. PWMCTL Field Descriptions**

| Field | Description |
|---|---|
| 7<br>CON67 | **Concatenate Channels 6 and 7**<br>0 Channels 6 and 7 are separate 8-bit PWMs.<br>1 Channels 6 and 7 are concatenated to create one 16-bit PWM channel. Channel 6 becomes the high order byte and channel 7 becomes the low order byte. Channel 7 output pin is used as the output for this 16-bit PWM (bit 7 of port PWMP). Channel 7 clock select control-bit determines the clock source, channel 7 polarity bit determines the polarity, channel 7 enable bit enables the output and channel 7 center aligned enable bit determines the output mode. |
| 6<br>CON45 | **Concatenate Channels 4 and 5**<br>0 Channels 4 and 5 are separate 8-bit PWMs.<br>1 Channels 4 and 5 are concatenated to create one 16-bit PWM channel. Channel 4 becomes the high order byte and channel 5 becomes the low order byte. Channel 5 output pin is used as the output for this 16-bit PWM (bit 5 of port PWMP). Channel 5 clock select control-bit determines the clock source, channel 5 polarity bit determines the polarity, channel 5 enable bit enables the output and channel 5 center aligned enable bit determines the output mode. |
| 5<br>CON23 | **Concatenate Channels 2 and 3**<br>0 Channels 2 and 3 are separate 8-bit PWMs.<br>1 Channels 2 and 3 are concatenated to create one 16-bit PWM channel. Channel 2 becomes the high order byte and channel 3 becomes the low order byte. Channel 3 output pin is used as the output for this 16-bit PWM (bit 3 of port PWMP). Channel 3 clock select control-bit determines the clock source, channel 3 polarity bit determines the polarity, channel 3 enable bit enables the output and channel 3 center aligned enable bit determines the output mode. |
| 4<br>CON01 | **Concatenate Channels 0 and 1**<br>0 Channels 0 and 1 are separate 8-bit PWMs.<br>1 Channels 0 and 1 are concatenated to create one 16-bit PWM channel. Channel 0 becomes the high order byte and channel 1 becomes the low order byte. Channel 1 output pin is used as the output for this 16-bit PWM (bit 1 of port PWMP). Channel 1 clock select control-bit determines the clock source, channel 1 polarity bit determines the polarity, channel 1 enable bit enables the output and channel 1 center aligned enable bit determines the output mode. |
| 3<br>PSWAI | **PWM Stops in Wait Mode —** Enabling this bit allows for lower power consumption in wait mode by disabling the input clock to the prescaler.<br>0 Allow the clock to the prescaler to continue while in wait mode.<br>1 Stop the input clock to the prescaler whenever the MCU is in wait mode. |
| 2<br>PFREZ | **PWM Counters Stop in Freeze Mode** — In freeze mode, there is an option to disable the input clock to the prescaler by setting the PFRZ bit in the PWMCTL register. If this bit is set, whenever the MCU is in freeze mode, the input clock to the prescaler is disabled. This feature is useful during emulation as it allows the PWM function to be suspended. In this way, the counters of the PWM can be stopped while in freeze mode so that once normal program flow is continued, the counters are re-enabled to simulate real-time operations. Since the registers can still be accessed in this mode, to re-enable the prescaler clock, either disable the PFRZ bit or exit freeze mode.<br>0 Allow PWM to continue while in freeze mode.<br>1 Disable PWM input clock to the prescaler whenever the part is in freeze mode. This is useful for emulation. |

### 19.3.2.7 Reserved Register (PWMTST)

This register is reserved for factory testing of the PWM module and is not available in normal modes.

Module Base + 0x0006

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| W | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented or Reserved

**Figure 19-9. Reserved Register (PWMTST)**

Read: Always read $00 in normal modes

Write: Unimplemented in normal modes

**NOTE**

Writing to this register when in special modes can alter the PWM functionality.

### 19.3.2.8 Reserved Register (PWMPRSC)

This register is reserved for factory testing of the PWM module and is not available in normal modes.

Module Base + 0x0007

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| W | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented or Reserved

**Figure 19-10. Reserved Register (PWMPRSC)**

Read: Always read $00 in normal modes

Write: Unimplemented in normal modes

**NOTE**

Writing to this register when in special modes can alter the PWM functionality.

### 19.3.2.9 PWM Scale A Register (PWMSCLA)

PWMSCLA is the programmable scale value used in scaling clock A to generate clock SA. Clock SA is generated by taking clock A, dividing it by the value in the PWMSCLA register and dividing that by two.

Clock SA = Clock A / (2 * PWMSCLA)

**NOTE**

When PWMSCLA = $00, PWMSCLA value is considered a full scale value of 256. Clock A is thus divided by 512.

Any value written to this register will cause the scale counter to load the new scale value (PWMSCLA).

Module Base + 0x0008

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 19-11. PWM Scale A Register (PWMSCLA)**

Read: Anytime

Write: Anytime (causes the scale counter to load the PWMSCLA value)

### 19.3.2.10 PWM Scale B Register (PWMSCLB)

PWMSCLB is the programmable scale value used in scaling clock B to generate clock SB. Clock SB is generated by taking clock B, dividing it by the value in the PWMSCLB register and dividing that by two.

Clock SB = Clock B / (2 * PWMSCLB)

**NOTE**

When PWMSCLB = $00, PWMSCLB value is considered a full scale value of 256. Clock B is thus divided by 512.

Any value written to this register will cause the scale counter to load the new scale value (PWMSCLB).

Module Base + 0x0009

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 19-12. PWM Scale B Register (PWMSCLB)**

Read: Anytime

Write: Anytime (causes the scale counter to load the PWMSCLB value).

### 19.3.2.11 Reserved Registers (PWMSCNTx)

The registers PWMSCNTA and PWMSCNTB are reserved for factory testing of the PWM module and are not available in normal modes.

Module Base + 0x000A, 0x000B

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| W | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented or Reserved

**Figure 19-13. Reserved Registers (PWMSCNTx)**

Read: Always read $00 in normal modes

Write: Unimplemented in normal modes

**NOTE**

Writing to these registers when in special modes can alter the PWM functionality.

### 19.3.2.12 PWM Channel Counter Registers (PWMCNTx)

Each channel has a dedicated 8-bit up/down counter which runs at the rate of the selected clock source. The counter can be read at any time without affecting the count or the operation of the PWM channel. In left aligned output mode, the counter counts from 0 to the value in the period register - 1. In center aligned output mode, the counter counts from 0 up to the value in the period register and then back down to 0.

Any value written to the counter causes the counter to reset to $00, the counter direction to be set to up, the immediate load of both duty and period registers with values from the buffers, and the output to change according to the polarity bit. The counter is also cleared at the end of the effective period (see Section 19.4.2.5, “Left Aligned Outputs” and Section 19.4.2.6, “Center Aligned Outputs” for more details). When the channel is disabled (PWMEx = 0), the PWMCNTx register does not count. When a channel becomes enabled (PWMEx = 1), the associated PWM counter starts at the count in the PWMCNTx register. For more detailed information on the operation of the counters, see Section 19.4.2.4, “PWM Timer Counters”.

In concatenated mode, writes to the 16-bit counter by using a 16-bit access or writes to either the low or high order byte of the counter will reset the 16-bit counter. Reads of the 16-bit counter must be made by 16-bit access to maintain data coherency.

**NOTE**

Writing to the counter while the channel is enabled can cause an irregular PWM cycle to occur.

Module Base + 0x000C = PWMCNT0, 0x000D = PWMCNT1, 0x000E = PWMCNT2, 0x000F = PWMCNT3
Module Base + 0x0010 = PWMCNT4, 0x0011 = PWMCNT5, 0x0012 = PWMCNT6, 0x0013 = PWMCNT7

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
| W | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 19-14. PWM Channel Counter Registers (PWMCNTx)**

Read: Anytime

Write: Anytime (any value written causes PWM counter to be reset to $00).

### 19.3.2.13 PWM Channel Period Registers (PWMPERx)

There is a dedicated period register for each channel. The value in this register determines the period of the associated PWM channel.

The period registers for each channel are double buffered so that if they change while the channel is enabled, the change will NOT take effect until one of the following occurs:

- The effective period ends
- The counter is written (counter resets to $00)
- The channel is disabled

In this way, the output of the PWM will always be either the old waveform or the new waveform, not some variation in between. If the channel is not enabled, then writes to the period register will go directly to the latches as well as the buffer.

**NOTE**

Reads of this register return the most recent value written. Reads do not necessarily return the value of the currently active period due to the double buffering scheme.

See Section 19.4.2.3, “PWM Period and Duty” for more information.

To calculate the output period, take the selected clock source period for the channel of interest (A, B, SA, or SB) and multiply it by the value in the period register for that channel:

- Left aligned output (CAEx = 0)
- PWMx Period = Channel Clock Period * PWMPERx Center Aligned Output (CAEx = 1)

  PWMx Period = Channel Clock Period * (2 * PWMPERx)

For boundary case programming values, please refer to Section 19.4.2.8, “PWM Boundary Cases”.

Module Base + 0x0014 = PWMPER0, 0x0015 = PWMPER1, 0x0016 = PWMPER2, 0x0017 = PWMPER3
Module Base + 0x0018 = PWMPER4, 0x0019 = PWMPER5, 0x001A = PWMPER6, 0x001B = PWMPER7

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
| Reset | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

**Figure 19-15. PWM Channel Period Registers (PWMPERx)**

Read: Anytime

Write: Anytime

### 19.3.2.14 PWM Channel Duty Registers (PWMDTYx)

There is a dedicated duty register for each channel. The value in this register determines the duty of the associated PWM channel. The duty value is compared to the counter and if it is equal to the counter value a match occurs and the output changes state.

The duty registers for each channel are double buffered so that if they change while the channel is enabled, the change will NOT take effect until one of the following occurs:

- The effective period ends
- The counter is written (counter resets to $00)
- The channel is disabled

In this way, the output of the PWM will always be either the old duty waveform or the new duty waveform, not some variation in between. If the channel is not enabled, then writes to the duty register will go directly to the latches as well as the buffer.

**NOTE**

Reads of this register return the most recent value written. Reads do not necessarily return the value of the currently active duty due to the double buffering scheme.

See Section 19.4.2.3, “PWM Period and Duty” for more information.

**NOTE**

Depending on the polarity bit, the duty registers will contain the count of either the high time or the low time. If the polarity bit is one, the output starts high and then goes low when the duty count is reached, so the duty registers contain a count of the high time. If the polarity bit is zero, the output starts low and then goes high when the duty count is reached, so the duty registers contain a count of the low time.

To calculate the output duty cycle (high time as a% of period) for a particular channel:

- Polarity = 0 (PPOL x =0)

  Duty Cycle = [(PWMPERx-PWMDTYx)/PWMPERx] * 100%
- Polarity = 1 (PPOLx = 1)

  Duty Cycle = [PWMDTYx / PWMPERx] * 100%

For boundary case programming values, please refer to Section 19.4.2.8, “PWM Boundary Cases”.

Module Base + 0x001C = PWMDTY0, 0x001D = PWMDTY1, 0x001E = PWMDTY2, 0x001F = PWMDTY3
Module Base + 0x0020 = PWMDTY4, 0x0021 = PWMDTY5, 0x0022 = PWMDTY6, 0x0023 = PWMDTY7

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
| Reset | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

**Figure 19-16. PWM Channel Duty Registers (PWMDTYx)**

Read: Anytime

Write: Anytime

### 19.3.2.15 PWM Shutdown Register (PWMSDN)

The PWMSDN register provides for the shutdown functionality of the PWM module in the emergency cases. For proper operation, channel 7 must be driven to the active level for a minimum of two bus clocks.

Module Base + 0x0024

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | PWMIF | PWMIE | 0 | PWMLVL | 0 | PWM7IN | PWM7INL | PWM7ENA |
| W | | | PWMRSTRT | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented or Reserved

**Figure 19-17. PWM Shutdown Register (PWMSDN)**

Read: Anytime

Write: Anytime

**Table 19-10. PWMSDN Field Descriptions**

| Field | Description |
|---|---|
| 7<br>PWMIF | **PWM Interrupt Flag** — Any change from passive to asserted (active) state or from active to passive state will be flagged by setting the PWMIF flag = 1. The flag is cleared by writing a logic 1 to it. Writing a 0 has no effect.<br>0 No change on PWM7IN input.<br>1 Change on PWM7IN input |
| 6<br>PWMIE | **PWM Interrupt Enable** — If interrupt is enabled an interrupt to the CPU is asserted.<br>0 PWM interrupt is disabled.<br>1 PWM interrupt is enabled. |
| 5<br>PWMRSTRT | **PWM Restart** — The PWM can only be restarted if the PWM channel input 7 is de-asserted. After writing a logic 1 to the PWMRSTRT bit (trigger event) the PWM channels start running after the corresponding counter passes next “counter == 0” phase. Also, if the PWM7ENA bit is reset to 0, the PWM do not start before the counter passes $00. The bit is always read as “0”. |
| 4<br>PWMLVL | **PWM Shutdown Output Level** If active level as defined by the PWM7IN input, gets asserted all enabled PWM channels are immediately driven to the level defined by PWMLVL.<br>0 PWM outputs are forced to 0<br>1 Outputs are forced to 1. |
| 2<br>PWM7IN | **PWM Channel 7 Input Status** — This reflects the current status of the PWM7 pin. |
| 1<br>PWM7INL | **PWM Shutdown Active Input Level for Channel 7** — If the emergency shutdown feature is enabled (PWM7ENA = 1), this bit determines the active level of the PWM7channel.<br>0 Active level is low<br>1 Active level is high |
| 0<br>PWM7ENA | **PWM Emergency Shutdown Enable** — If this bit is logic 1, the pin associated with channel 7 is forced to input and the emergency shutdown feature is enabled. All the other bits in this register are meaningful only if PWM7ENA = 1.<br>0 PWM emergency feature disabled.<br>1 PWM emergency feature is enabled. |

# 19.4 Functional Description

## 19.4.1 PWM Clock Select

There are four available clocks: clock A, clock B, clock SA (scaled A), and clock SB (scaled B). These four clocks are based on the bus clock.

Clock A and B can be software selected to be 1, 1/2, 1/4, 1/8,..., 1/64, 1/128 times the bus clock. Clock SA uses clock A as an input and divides it further with a reloadable counter. Similarly, clock SB uses clock B as an input and divides it further with a reloadable counter. The rates available for clock SA are software selectable to be clock A divided by 2, 4, 6, 8,..., or 512 in increments of divide by 2. Similar rates are available for clock SB. Each PWM channel has the capability of selecting one of two clocks, either the pre-scaled clock (clock A or B) or the scaled clock (clock SA or SB).

The block diagram in Figure 19-18 shows the four different clocks and how the scaled clocks are created.

### 19.4.1.1 Prescale

The input clock to the PWM prescaler is the bus clock. It can be disabled whenever the part is in freeze mode by setting the PFRZ bit in the PWMCTL register. If this bit is set, whenever the MCU is in freeze mode (freeze mode signal active) the input clock to the prescaler is disabled. This is useful for emulation in order to freeze the PWM. The input clock can also be disabled when all eight PWM channels are disabled (PWME7-0 = 0). This is useful for reducing power by disabling the prescale counter.

Clock A and clock B are scaled values of the input clock. The value is software selectable for both clock A and clock B and has options of 1, 1/2, 1/4, 1/8, 1/16, 1/32, 1/64, or 1/128 times the bus clock. The value selected for clock A is determined by the PCKA2, PCKA1, PCKA0 bits in the PWMPRCLK register. The value selected for clock B is determined by the PCKB2, PCKB1, PCKB0 bits also in the PWMPRCLK register.

### 19.4.1.2 Clock Scale

The scaled A clock uses clock A as an input and divides it further with a user programmable value and then divides this by 2. The scaled B clock uses clock B as an input and divides it further with a user programmable value and then divides this by 2. The rates available for clock SA are software selectable to be clock A divided by 2, 4, 6, 8,..., or 512 in increments of divide by 2. Similar rates are available for clock SB.

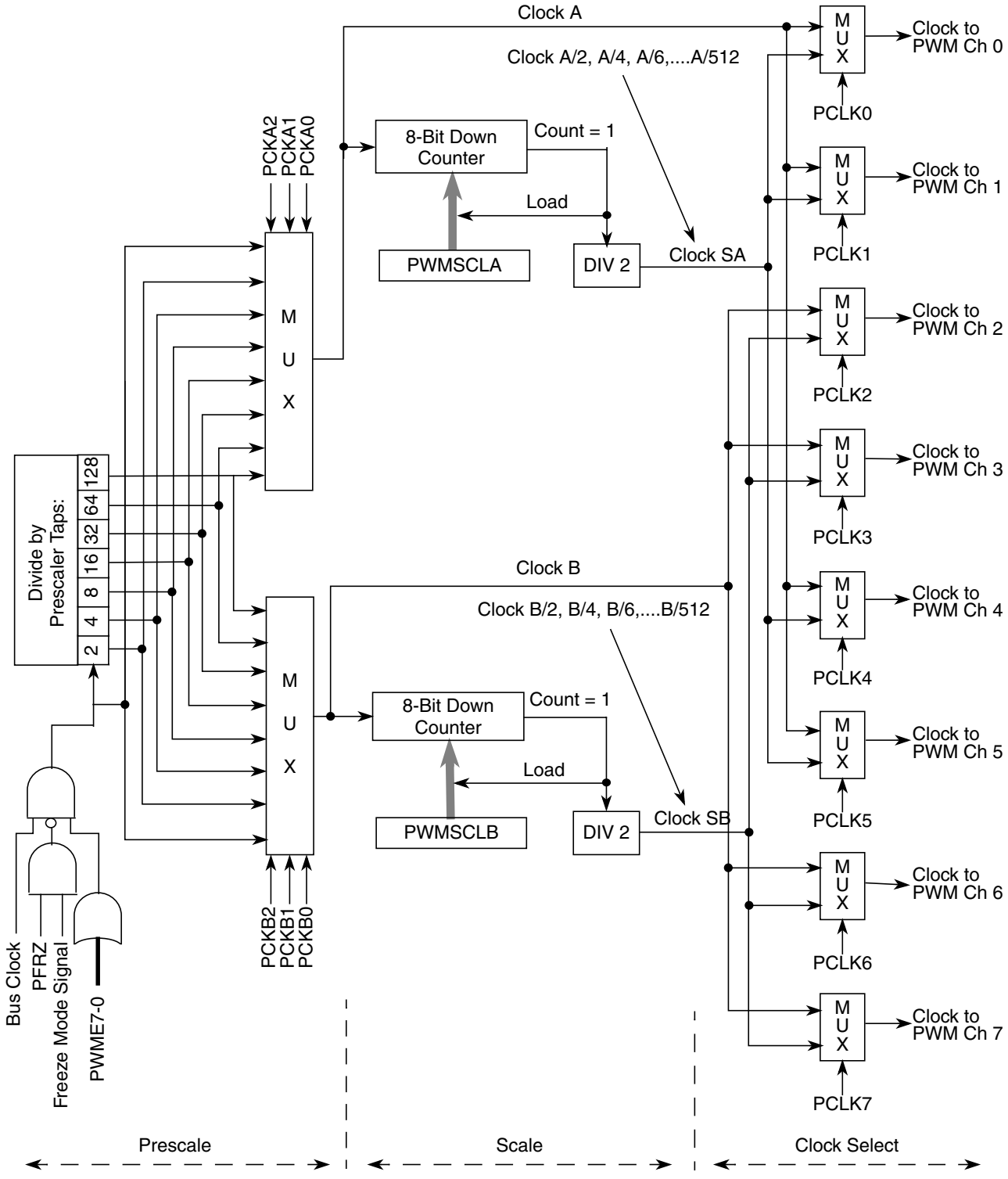



**Figure 19-18. PWM Clock Select Block Diagram**

Clock A is used as an input to an 8-bit down counter. This down counter loads a user programmable scale value from the scale register (PWMSCLA). When the down counter reaches one, a pulse is output and the 8-bit counter is re-loaded. The output signal from this circuit is further divided by two. This gives a greater range with only a slight reduction in granularity. Clock SA equals clock A divided by two times the value in the PWMSCLA register.

**NOTE**

Clock SA = Clock A / (2 * PWMSCLA)

When PWMSCLA = $00, PWMSCLA value is considered a full scale value of 256. Clock A is thus divided by 512.

Similarly, clock B is used as an input to an 8-bit down counter followed by a divide by two producing clock SB. Thus, clock SB equals clock B divided by two times the value in the PWMSCLB register.

**NOTE**

Clock SB = Clock B / (2 * PWMSCLB)

When PWMSCLB = $00, PWMSCLB value is considered a full scale value of 256. Clock B is thus divided by 512.

As an example, consider the case in which the user writes $FF into the PWMSCLA register. Clock A for this case will be E divided by 4. A pulse will occur at a rate of once every 255x4 E cycles. Passing this through the divide by two circuit produces a clock signal at an E divided by 2040 rate. Similarly, a value of $01 in the PWMSCLA register when clock A is E divided by 4 will produce a clock at an E divided by 8 rate.

Writing to PWMSCLA or PWMSCLB causes the associated 8-bit down counter to be re-loaded. Otherwise, when changing rates the counter would have to count down to $01 before counting at the proper rate. Forcing the associated counter to re-load the scale register value every time PWMSCLA or PWMSCLB is written prevents this.

**NOTE**

Writing to the scale registers while channels are operating can cause irregularities in the PWM outputs.

#### 19.4.1.3 Clock Select

Each PWM channel has the capability of selecting one of two clocks. For channels 0, 1, 4, and 5 the clock choices are clock A or clock SA. For channels 2, 3, 6, and 7 the choices are clock B or clock SB. The clock selection is done with the PCLKx control bits in the PWMCLK register.

**NOTE**

Changing clock control bits while channels are operating can cause irregularities in the PWM outputs.

## 19.4.2 PWM Channel Timers

The main part of the PWM module are the actual timers. Each of the timer channels has a counter, a period register and a duty register (each are 8-bit). The waveform output period is controlled by a match between the period register and the value in the counter. The duty is controlled by a match between the duty register and the counter value and causes the state of the output to change during the period. The starting polarity of the output is also selectable on a per channel basis. Shown below in Figure 19-19 is the block diagram for the PWM timer.

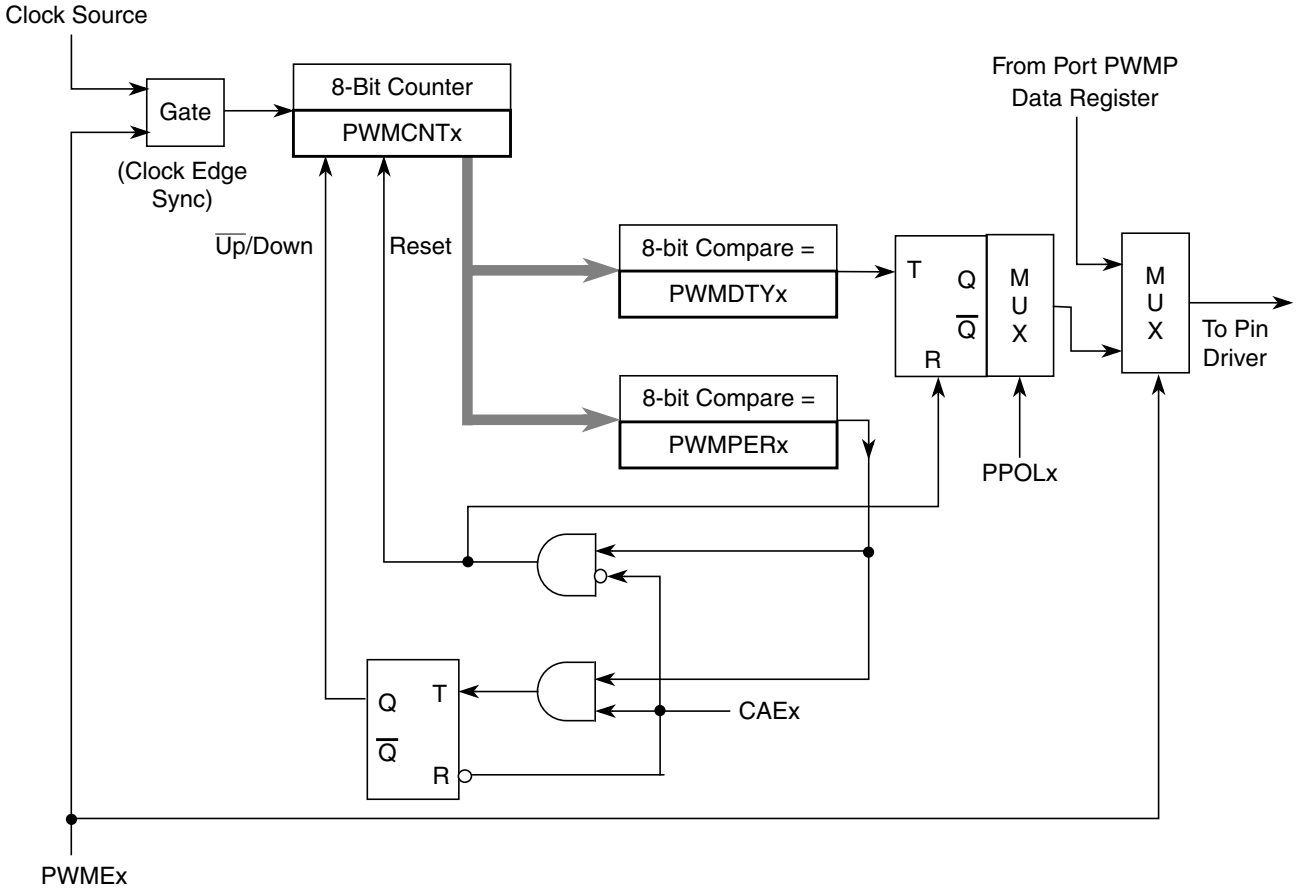



**Figure 19-19. PWM Timer Channel Block Diagram**

### 19.4.2.1 PWM Enable

Each PWM channel has an enable bit (PWMEx) to start its waveform output. When any of the PWMEx bits are set (PWMEx = 1), the associated PWM output signal is enabled immediately. However, the actual PWM waveform is not available on the associated PWM output until its clock source begins its next cycle due to the synchronization of PWMEx and the clock source. An exception to this is when channels are concatenated. Refer to Section 19.4.2.7, “PWM 16-Bit Functions” for more detail.

**NOTE**

The first PWM cycle after enabling the channel can be irregular.

On the front end of the PWM timer, the clock is enabled to the PWM circuit by the PWMEx bit being high. There is an edge-synchronizing circuit to guarantee that the clock will only be enabled or disabled at an edge. When the channel is disabled (PWMEx = 0), the counter for the channel does not count.

### 19.4.2.2 PWM Polarity

Each channel has a polarity bit to allow starting a waveform cycle with a high or low signal. This is shown on the block diagram as a mux select of either the Q output or the $\overline{Q}$ output of the PWM output flip flop. When one of the bits in the PWMPOL register is set, the associated PWM channel output is high at the beginning of the waveform, then goes low when the duty count is reached. Conversely, if the polarity bit is zero, the output starts low and then goes high when the duty count is reached.

### 19.4.2.3 PWM Period and Duty

Dedicated period and duty registers exist for each channel and are double buffered so that if they change while the channel is enabled, the change will NOT take effect until one of the following occurs:

- The effective period ends
- The counter is written (counter resets to $00)
- The channel is disabled

In this way, the output of the PWM will always be either the old waveform or the new waveform, not some variation in between. If the channel is not enabled, then writes to the period and duty registers will go directly to the latches as well as the buffer.

A change in duty or period can be forced into effect “immediately” by writing the new value to the duty and/or period registers and then writing to the counter. This forces the counter to reset and the new duty and/or period values to be latched. In addition, since the counter is readable, it is possible to know where the count is with respect to the duty value and software can be used to make adjustments

**NOTE**

When forcing a new period or duty into effect immediately, an irregular PWM cycle can occur.

Depending on the polarity bit, the duty registers will contain the count of either the high time or the low time.

### 19.4.2.4 PWM Timer Counters

Each channel has a dedicated 8-bit up/down counter which runs at the rate of the selected clock source (see Section 19.4.1, “PWM Clock Select” for the available clock sources and rates). The counter compares to two registers, a duty register and a period register as shown in Figure 19-19. When the PWM counter matches the duty register, the output flip-flop changes state, causing the PWM waveform to also change state. A match between the PWM counter and the period register behaves differently depending on what output mode is selected as shown in Figure 19-19 and described in Section 19.4.2.5, “Left Aligned Outputs” and Section 19.4.2.6, “Center Aligned Outputs”.

Each channel counter can be read at anytime without affecting the count or the operation of the PWM channel.

Any value written to the counter causes the counter to reset to $00, the counter direction to be set to up, the immediate load of both duty and period registers with values from the buffers, and the output to change according to the polarity bit. When the channel is disabled (PWMEx = 0), the counter stops. When a channel becomes enabled (PWMEx = 1), the associated PWM counter continues from the count in the PWMCNTx register. This allows the waveform to continue where it left off when the channel is re-enabled. When the channel is disabled, writing “0” to the period register will cause the counter to reset on the next selected clock.

**NOTE**

If the user wants to start a new “clean” PWM waveform without any “history” from the old waveform, the user must write to channel counter (PWMCNTx) prior to enabling the PWM channel (PWMEx = 1).

Generally, writes to the counter are done prior to enabling a channel in order to start from a known state. However, writing a counter can also be done while the PWM channel is enabled (counting). The effect is similar to writing the counter when the channel is disabled, except that the new period is started immediately with the output set according to the polarity bit.

**NOTE**

Writing to the counter while the channel is enabled can cause an irregular PWM cycle to occur.

The counter is cleared at the end of the effective period (see Section 19.4.2.5, “Left Aligned Outputs” and Section 19.4.2.6, “Center Aligned Outputs” for more details).

**Table 19-11. PWM Timer Counter Conditions**

| Counter Clears ($00) | Counter Counts | Counter Stops |
|---|---|---|
| When PWMCNTx register written to any value | When PWM channel is enabled (PWMEx = 1). Counts from last value in PWMCNTx. | When PWM channel is disabled (PWMEx = 0) |
| Effective period ends | | |

### 19.4.2.5 Left Aligned Outputs

The PWM timer provides the choice of two types of outputs, left aligned or center aligned. They are selected with the CAEx bits in the PWMCAE register. If the CAEx bit is cleared (CAEx = 0), the corresponding PWM output will be left aligned.

In left aligned output mode, the 8-bit counter is configured as an up counter only. It compares to two registers, a duty register and a period register as shown in the block diagram in Figure 19-19. When the PWM counter matches the duty register the output flip-flop changes state causing the PWM waveform to also change state. A match between the PWM counter and the period register resets the counter and the output flip-flop, as shown in Figure 19-19, as well as performing a load from the double buffer period and duty register to the associated registers, as described in Section 19.4.2.3, “PWM Period and Duty”. The counter counts from 0 to the value in the period register – 1.

**NOTE**

Changing the PWM output mode from left aligned to center aligned output (or vice versa) while channels are operating can cause irregularities in the PWM output. It is recommended to program the output mode before enabling the PWM channel.

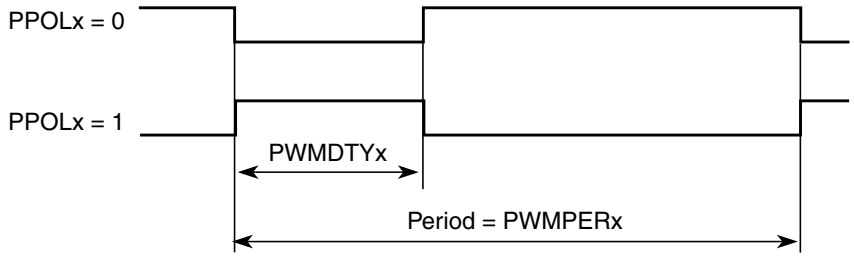



**Figure 19-20. PWM Left Aligned Output Waveform**

To calculate the output frequency in left aligned output mode for a particular channel, take the selected clock source frequency for the channel (A, B, SA, or SB) and divide it by the value in the period register for that channel.

- PWMx Frequency = Clock (A, B, SA, or SB) / PWMPERx
- PWMx Duty Cycle (high time as a% of period):
  - Polarity = 0 (PPOLx = 0)
- Duty Cycle = [(PWMPERx-PWMDTYx)/PWMPERx] * 100%
  - Polarity = 1 (PPOLx = 1)

Duty Cycle = [PWMDTYx / PWMPERx] * 100%

As an example of a left aligned output, consider the following case:

Clock Source = E, where E = 10 MHz (100 ns period)

PPOLx = 0

PWMPERx = 4

PWMDTYx = 1

PWMx Frequency = 10 MHz/4 = 2.5 MHz

PWMx Period = 400 ns

PWMx Duty Cycle = 3/4 *100% = 75%

The output waveform generated is shown in Figure 19-21.

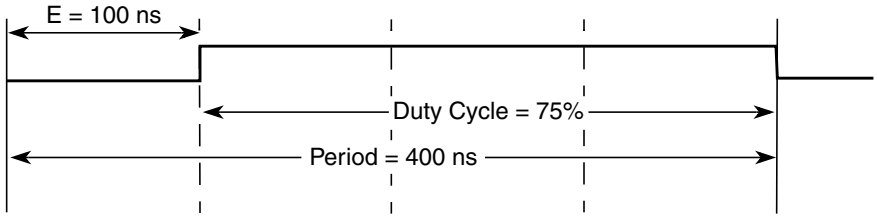



**Figure 19-21. PWM Left Aligned Output Example Waveform**

### 19.4.2.6 Center Aligned Outputs

For center aligned output mode selection, set the CAEx bit (CAEx = 1) in the PWMCAE register and the corresponding PWM output will be center aligned.

The 8-bit counter operates as an up/down counter in this mode and is set to up whenever the counter is equal to $00. The counter compares to two registers, a duty register and a period register as shown in the block diagram in Figure 19-19. When the PWM counter matches the duty register, the output flip-flop changes state, causing the PWM waveform to also change state. A match between the PWM counter and the period register changes the counter direction from an up-count to a down-count. When the PWM counter decrements and matches the duty register again, the output flip-flop changes state causing the PWM output to also change state. When the PWM counter decrements and reaches zero, the counter direction changes from a down-count back to an up-count and a load from the double buffer period and duty registers to the associated registers is performed, as described in Section 19.4.2.3, “PWM Period and Duty”. The counter counts from 0 up to the value in the period register and then back down to 0. Thus the effective period is PWMPERx*2.

**NOTE**

Changing the PWM output mode from left aligned to center aligned output (or vice versa) while channels are operating can cause irregularities in the PWM output. It is recommended to program the output mode before enabling the PWM channel.

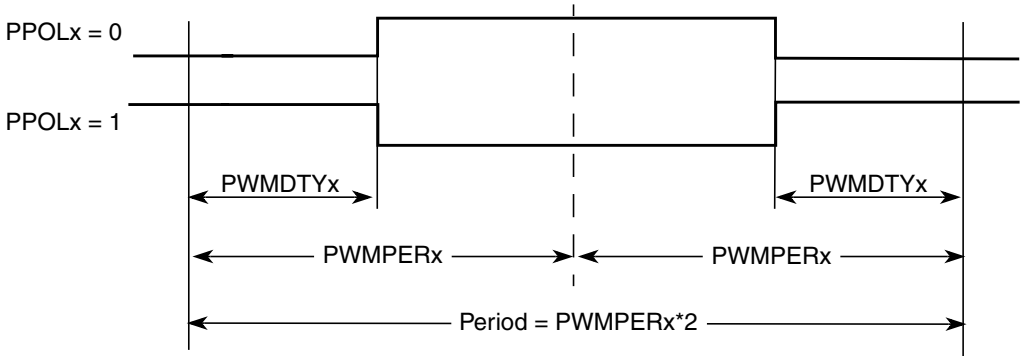



**Figure 19-22. PWM Center Aligned Output Waveform**

To calculate the output frequency in center aligned output mode for a particular channel, take the selected clock source frequency for the channel (A, B, SA, or SB) and divide it by twice the value in the period register for that channel.

- PWMx Frequency = Clock (A, B, SA, or SB) / (2*PWMPERx)
- PWMx Duty Cycle (high time as a% of period):
  - — Polarity = 0 (PPOLx = 0)

    Duty Cycle = [(PWMPERx-PWMDTYx)/PWMPERx] * 100%
  - — Polarity = 1 (PPOLx = 1)

    Duty Cycle = [PWMDTYx / PWMPERx] * 100%

As an example of a center aligned output, consider the following case:

Clock Source = E, where E = 10 MHz (100 ns period)

PPOLx = 0

PWMPERx = 4

PWMDTYx = 1

PWMx Frequency = 10 MHz/8 = 1.25 MHz

PWMx Period = 800 ns

PWMx Duty Cycle = 3/4 *100% = 75%

Shown in Figure 19-23 is the output waveform generated.

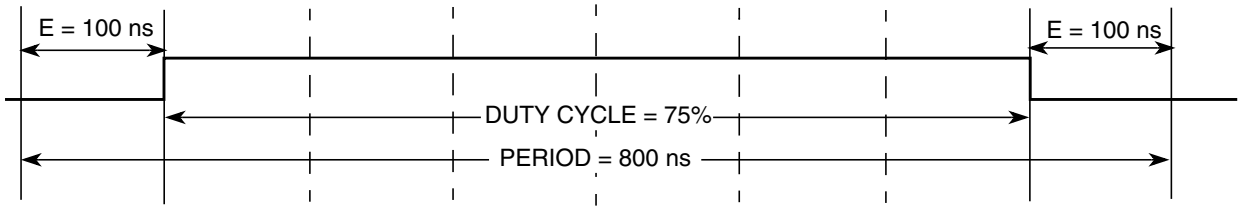



**Figure 19-23. PWM Center Aligned Output Example Waveform**

### 19.4.2.7 PWM 16-Bit Functions

The PWM timer also has the option of generating 8-channels of 8-bits or 4-channels of 16-bits for greater PWM resolution. This 16-bit channel option is achieved through the concatenation of two 8-bit channels.

The PWMCTL register contains four control bits, each of which is used to concatenate a pair of PWM channels into one 16-bit channel. Channels 6 and 7 are concatenated with the CON67 bit, channels 4 and 5 are concatenated with the CON45 bit, channels 2 and 3 are concatenated with the CON23 bit, and channels 0 and 1 are concatenated with the CON01 bit.

**NOTE**

Change these bits only when both corresponding channels are disabled.

When channels 6 and 7 are concatenated, channel 6 registers become the high order bytes of the double byte channel, as shown in Figure 19-24. Similarly, when channels 4 and 5 are concatenated, channel 4 registers become the high order bytes of the double byte channel. When channels 2 and 3 are concatenated,

channel 2 registers become the high order bytes of the double byte channel. When channels 0 and 1 are concatenated, channel 0 registers become the high order bytes of the double byte channel.

When using the 16-bit concatenated mode, the clock source is determined by the low order 8-bit channel clock select control bits. That is channel 7 when channels 6 and 7 are concatenated, channel 5 when channels 4 and 5 are concatenated, channel 3 when channels 2 and 3 are concatenated, and channel 1 when channels 0 and 1 are concatenated. The resulting PWM is output to the pins of the corresponding low order 8-bit channel as also shown in Figure 19-24. The polarity of the resulting PWM output is controlled by the PPOLx bit of the corresponding low order 8-bit channel as well.

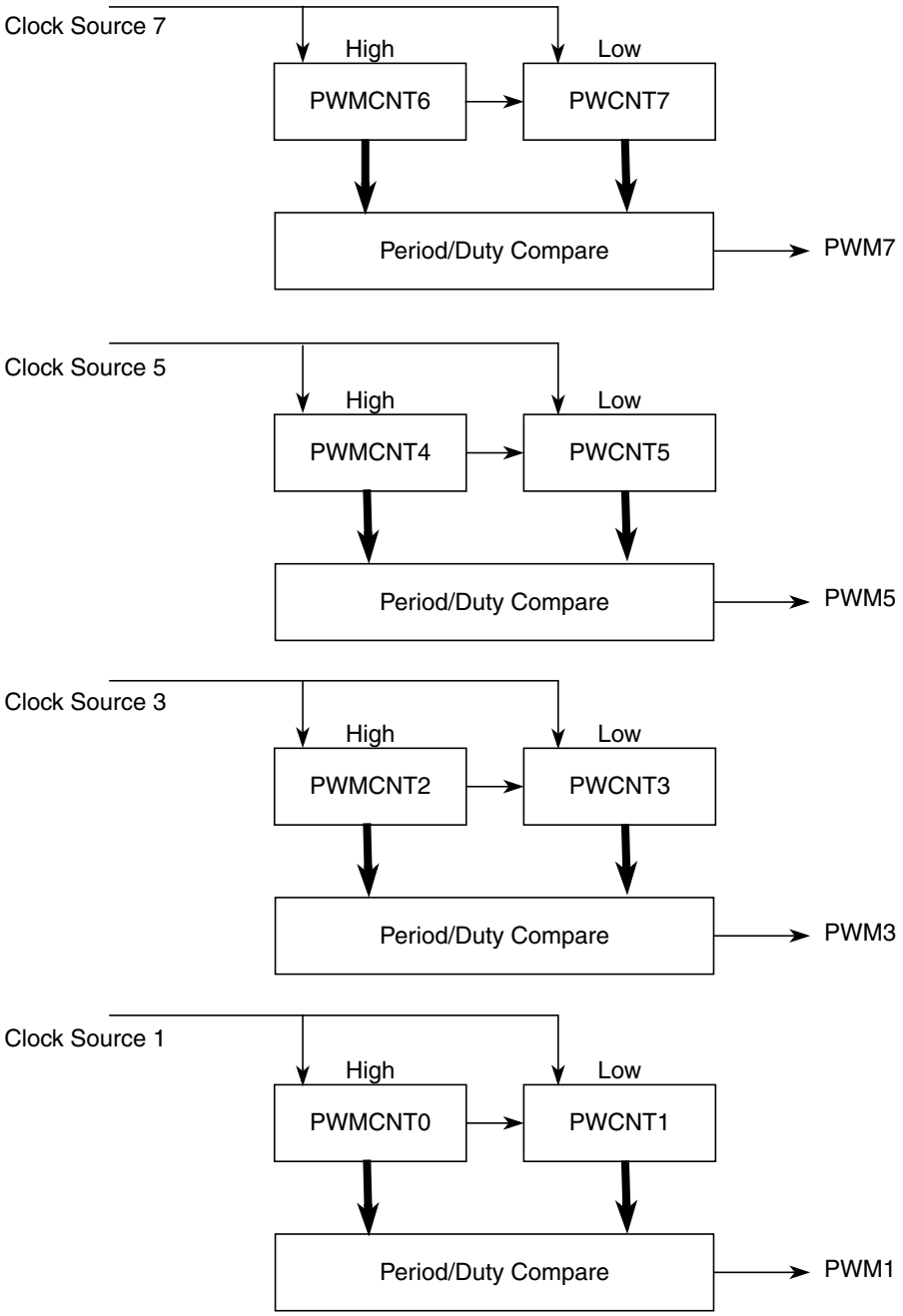



**Figure 19-24. PWM 16-Bit Mode**

Once concatenated mode is enabled (CONxx bits set in PWMCTL register), enabling/disabling the corresponding 16-bit PWM channel is controlled by the low order PWMEx bit. In this case, the high order bytes PWMEx bits have no effect and their corresponding PWM output is disabled.

In concatenated mode, writes to the 16-bit counter by using a 16-bit access or writes to either the low or high order byte of the counter will reset the 16-bit counter. Reads of the 16-bit counter must be made by 16-bit access to maintain data coherency.

Either left aligned or center aligned output mode can be used in concatenated mode and is controlled by the low order CAEx bit. The high order CAEx bit has no effect.

Table 19-12 is used to summarize which channels are used to set the various control bits when in 16-bit mode.

**Table 19-12. 16-bit Concatenation Mode Summary**

| CONxx | PWMEx | PPOLx | PCLKx | CAEx | PWMx Output |
|---|---|---|---|---|---|
| CON67 | PWME7 | PPOL7 | PCLK7 | CAE7 | PWM7 |
| CON45 | PWME5 | PPOL5 | PCLK5 | CAE5 | PWM5 |
| CON23 | PWME3 | PPOL3 | PCLK3 | CAE3 | PWM3 |
| CON01 | PWME1 | PPOL1 | PCLK1 | CAE1 | PWM1 |

### 19.4.2.8 PWM Boundary Cases

Table 19-13 summarizes the boundary conditions for the PWM regardless of the output mode (left aligned or center aligned) and 8-bit (normal) or 16-bit (concatenation).

**Table 19-13. PWM Boundary Cases**

| PWMDTYx | PWMPERx | PPOLx | PWMx Output |
|---|---|---|---|
| $00 (indicates no duty) | >$00 | 1 | Always low |
| $00 (indicates no duty) | >$00 | 0 | Always high |
| XX | $00[1] (indicates no period) | 1 | Always high |
| XX | $00[1] (indicates no period) | 0 | Always low |
| >= PWMPERx | XX | 1 | Always high |
| >= PWMPERx | XX | 0 | Always low |

1. Counter = $00 and does not count.

## 19.5 Resets

The reset state of each individual bit is listed within the Section 19.3.2, “Register Descriptions” which details the registers and their bit-fields. All special functions or modes which are initialized during or just following reset are described within this section.

- The 8-bit up/down counter is configured as an up counter out of reset.
- All the channels are disabled and all the counters do not count.

## 19.6 Interrupts

The PWM module has only one interrupt which is generated at the time of emergency shutdown, if the corresponding enable bit (PWMIE) is set. This bit is the enable for the interrupt. The interrupt flag PWMIF is set whenever the input level of the PWM7 channel changes while PWM7ENA = 1 or when PWMENA is being asserted while the level at PWM7 is active.

In stop mode or wait mode (with the PSWAI bit set), the emergency shutdown feature will drive the PWM outputs to their shutdown output levels but the PWMIF flag will not be set.

A description of the registers involved and affected due to this interrupt is explained in Section 19.3.2.15, “PWM Shutdown Register (PWMSDN)”.

The PWM block only generates the interrupt and does not service it. The interrupt signal name is PWM interrupt signal.

# Chapter 20
# Serial Communication Interface (S12SCIV5)

**Table 20-1. Revision History**

| Version Number | Revision Date | Effective Date | Author | Description of Changes |
|---|---|---|---|---|
| 05.03 | 12/25/2008 | | | remove redundancy comments in Figure1-2 |
| 05.04 | 08/05/2009 | | | fix typo, SCIBDL reset value be 0x04, not 0x00 |
| 05.05 | 06/03/2010 | | | fix typo, Table 20-4,SCICR1 Even parity should be PT=0<br>fix typo, on page 20-745,should be BKDIF,not BLDIF |

## 20.1 Introduction

This block guide provides an overview of the serial communication interface (SCI) module.

The SCI allows asynchronous serial communications with peripheral devices and other CPUs.

### 20.1.1 Glossary

IR: InfraRed

IrDA: Infrared Design Associate

IRQ: Interrupt Request

LIN: Local Interconnect Network

LSB: Least Significant Bit

MSB: Most Significant Bit

NRZ: Non-Return-to-Zero

RZI: Return-to-Zero-Inverted

RXD: Receive Pin

SCI : Serial Communication Interface

TXD: Transmit Pin

## 20.1.2 Features

The SCI includes these distinctive features:

- Full-duplex or single-wire operation
- Standard mark/space non-return-to-zero (NRZ) format
- Selectable IrDA 1.4 return-to-zero-inverted (RZI) format with programmable pulse widths
- 13-bit baud rate selection
- Programmable 8-bit or 9-bit data format
- Separately enabled transmitter and receiver
- Programmable polarity for transmitter and receiver
- Programmable transmitter output parity
- Two receiver wakeup methods:
  - Idle line wakeup
  - Address mark wakeup
- Interrupt-driven operation with eight flags:
  - Transmitter empty
  - Transmission complete
  - Receiver full
  - Idle receiver input
  - Receiver overrun
  - Noise error
  - Framing error
  - Parity error
  - Receive wakeup on active edge
  - Transmit collision detect supporting LIN
  - Break Detect supporting LIN
- Receiver framing error detection
- Hardware parity checking
- 1/16 bit-time noise detection

## 20.1.3 Modes of Operation

The SCI functions the same in normal, special, and emulation modes. It has two low power modes, wait and stop modes.

- Run mode
- Wait mode
- Stop mode

### 20.1.4 Block Diagram

Figure 20-1 is a high level block diagram of the SCI module, showing the interaction of various function blocks.

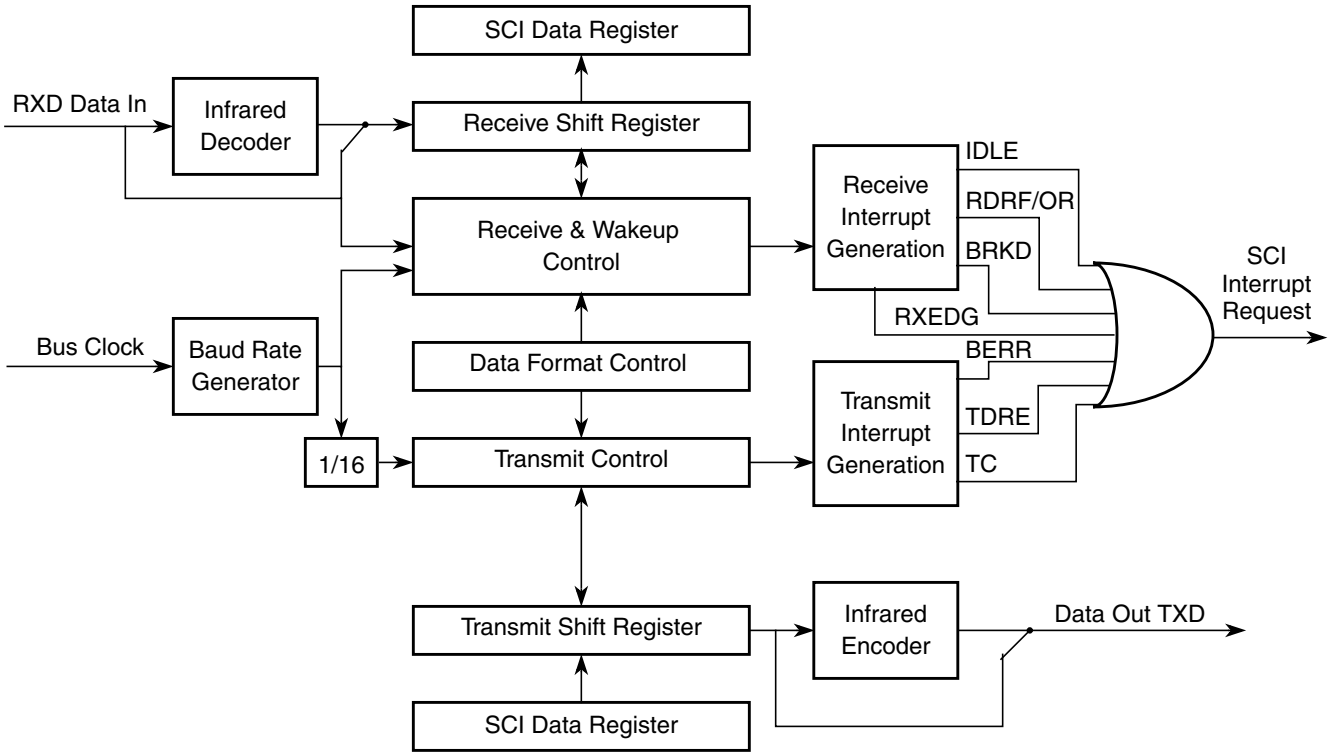



**Figure 20-1. SCI Block Diagram**

## 20.2 External Signal Description

The SCI module has a total of two external pins.

### 20.2.1 TXD — Transmit Pin

The TXD pin transmits SCI (standard or infrared) data. It will idle high in either mode and is high impedance anytime the transmitter is disabled.

### 20.2.2 RXD — Receive Pin

The RXD pin receives SCI (standard or infrared) data. An idle line is detected as a line high. This input is ignored when the receiver is disabled and should be terminated to a known voltage.

## 20.3 Memory Map and Register Definition

This section provides a detailed description of all the SCI registers.

### 20.3.1 Module Memory Map and Register Definition

The memory map for the SCI module is given below in Figure 20-2. The address listed for each register is the address offset. The total address for each register is the sum of the base address for the SCI module and the address offset for each register.

## 20.3.2 Register Descriptions

This section consists of register descriptions in address order. Each description includes a standard register diagram with an associated figure number. Writes to a reserved register locations do not have any effect and reads of these locations return a zero. Details of register bit and field function follow the register diagrams, in bit order.

| Register Name | | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|---|
| 0x0000 SCIBDH[1] | R | IREN | TNP1 | TNP0 | SBR12 | SBR11 | SBR10 | SBR9 | SBR8 |
| | W | | | | | | | | |
| 0x0001 SCIBDL[1] | R | SBR7 | SBR6 | SBR5 | SBR4 | SBR3 | SBR2 | SBR1 | SBR0 |
| | W | | | | | | | | |
| 0x0002 SCICR1[1] | R | LOOPS | SCISWAI | RSRC | M | WAKE | ILT | PE | PT |
| | W | | | | | | | | |
| 0x0000 SCIASR1[2] | R | RXEDGIF | 0 | 0 | 0 | 0 | BERRV | BERRIF | BKDIF |
| | W | | | | | | | | |
| 0x0001 SCIACR1[2] | R | RXEDGIE | 0 | 0 | 0 | 0 | 0 | BERRIE | BKDIE |
| | W | | | | | | | | |
| 0x0002 SCIACR2[2] | R | 0 | 0 | 0 | 0 | 0 | BERRM1 | BERRM0 | BKDFE |
| | W | | | | | | | | |
| 0x0003 SCICR2 | R | TIE | TCIE | RIE | ILIE | TE | RE | RWU | SBK |
| | W | | | | | | | | |
| 0x0004 SCISR1 | R | TDRE | TC | RDRF | IDLE | OR | NF | FE | PF |
| | W | | | | | | | | |
| 0x0005 SCISR2 | R | AMAP | 0 | 0 | TXPOL | RXPOL | BRK13 | TXDIR | RAF |
| | W | | | | | | | | |
| 0x0006 SCIDRH | R | R8 | T8 | 0 | 0 | 0 | 0 | 0 | 0 |
| | W | | | | | | | | |
| 0x0007 SCIDRL | R | R7 | R6 | R5 | R4 | R3 | R2 | R1 | R0 |
| | W | T7 | T6 | T5 | T4 | T3 | T2 | T1 | T0 |

1.These registers are accessible if the AMAP bit in the SCISR2 register is set to zero.

2,These registers are accessible if the AMAP bit in the SCISR2 register is set to one.

= Unimplemented or Reserved

**Figure 20-2. SCI Register Summary**

### 20.3.2.1 SCI Baud Rate Registers (SCIBDH, SCIBDL)

Module Base + 0x0000

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | IREN | TNP1 | TNP0 | SBR12 | SBR11 | SBR10 | SBR9 | SBR8 |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 20-3. SCI Baud Rate Register (SCIBDH)**

Module Base + 0x0001

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | SBR7 | SBR6 | SBR5 | SBR4 | SBR3 | SBR2 | SBR1 | SBR0 |
| Reset | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |

**Figure 20-4. SCI Baud Rate Register (SCIBDL)**

Read: Anytime, if AMAP = 0. If only SCIBDH is written to, a read will not return the correct data until SCIBDL is written to as well, following a write to SCIBDH.

Write: Anytime, if AMAP = 0.

**NOTE**

Those two registers are only visible in the memory map if AMAP = 0 (reset condition).

The SCI baud rate register is used by to determine the baud rate of the SCI, and to control the infrared modulation/demodulation submodule.

**Table 20-2. SCIBDH and SCIBDL Field Descriptions**

| Field | Description |
|---|---|
| 7<br>IREN | **Infrared Enable Bit** — This bit enables/disables the infrared modulation/demodulation submodule.<br>0 IR disabled<br>1 IR enabled |
| 6:5<br>TNP[1:0] | **Transmitter Narrow Pulse Bits** — These bits enable whether the SCI transmits a 1/16, 3/16, 1/32 or 1/4 narrow pulse. See Table 20-3. |
| 4:0<br>7:0<br>SBR[12:0] | **SCI Baud Rate Bits** — The baud rate for the SCI is determined by the bits in this register. The baud rate is calculated two different ways depending on the state of the IREN bit.<br>The formulas for calculating the baud rate are:<br>When IREN = 0 then,<br>SCI baud rate = SCI bus clock / (16 x SBR[12:0])<br>When IREN = 1 then,<br>SCI baud rate = SCI bus clock / (32 x SBR[12:1])<br>**Note:** The baud rate generator is disabled after reset and not started until the TE bit or the RE bit is set for the first time. The baud rate generator is disabled when (SBR[12:0] = 0 and IREN = 0) or (SBR[12:1] = 0 and IREN = 1).<br>**Note:** Writing to SCIBDH has no effect without writing to SCIBDL, because writing to SCIBDH puts the data in a temporary location until SCIBDL is written to. |

**Table 20-3. IRSCI Transmit Pulse Width**

| TNP[1:0] | Narrow Pulse Width |
|---|---|
| 11 | 1/4 |
| 10 | 1/32 |
| 01 | 1/16 |
| 00 | 3/16 |

### 20.3.2.2 SCI Control Register 1 (SCICR1)

Module Base + 0x0002

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | LOOPS | SCISWAI | RSRC | M | WAKE | ILT | PE | PT |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 20-5. SCI Control Register 1 (SCICR1)**

Read: Anytime, if AMAP = 0.

Write: Anytime, if AMAP = 0.

**NOTE**

This register is only visible in the memory map if AMAP = 0 (reset condition).

**Table 20-4. SCICR1 Field Descriptions**

| Field | Description |
|---|---|
| 7<br>LOOPS | **Loop Select Bit** — LOOPS enables loop operation. In loop operation, the RXD pin is disconnected from the SCI and the transmitter output is internally connected to the receiver input. Both the transmitter and the receiver must be enabled to use the loop function.<br>0 Normal operation enabled<br>1 Loop operation enabled<br>The receiver input is determined by the RSRC bit. |
| 6<br>SCISWAI | **SCI Stop in Wait Mode Bit** — SCISWAI disables the SCI in wait mode.<br>0 SCI enabled in wait mode<br>1 SCI disabled in wait mode |
| 5<br>RSRC | **Receiver Source Bit** — When LOOPS = 1, the RSRC bit determines the source for the receiver shift register input. See Table 20-5.<br>0 Receiver input internally connected to transmitter output<br>1 Receiver input connected externally to transmitter |
| 4<br>M | **Data Format Mode Bit** — MODE determines whether data characters are eight or nine bits long.<br>0 One start bit, eight data bits, one stop bit<br>1 One start bit, nine data bits, one stop bit |
| 3<br>WAKE | **Wakeup Condition Bit** — WAKE determines which condition wakes up the SCI: a logic 1 (address mark) in the most significant bit position of a received data character or an idle condition on the RXD pin.<br>0 Idle line wakeup<br>1 Address mark wakeup |

**Table 20-4. SCICR1 Field Descriptions (continued)**

| Field | Description |
|---|---|
| 2<br>ILT | **Idle Line Type Bit** — ILT determines when the receiver starts counting logic 1s as idle character bits. The counting begins either after the start bit or after the stop bit. If the count begins after the start bit, then a string of logic 1s preceding the stop bit may cause false recognition of an idle character. Beginning the count after the stop bit avoids false idle character recognition, but requires properly synchronized transmissions.<br>0 Idle character bit count begins after start bit<br>1 Idle character bit count begins after stop bit |
| 1<br>PE | **Parity Enable Bit** — PE enables the parity function. When enabled, the parity function inserts a parity bit in the most significant bit position.<br>0 Parity function disabled<br>1 Parity function enabled |
| 0<br>PT | **Parity Type Bit** — PT determines whether the SCI generates and checks for even parity or odd parity. With even parity, an even number of 1s clears the parity bit and an odd number of 1s sets the parity bit. With odd parity, an odd number of 1s clears the parity bit and an even number of 1s sets the parity bit.<br>0 Even parity<br>1 Odd parity |

**Table 20-5. Loop Functions**

| LOOPS | RSRC | Function |
|---|---|---|
| 0 | x | Normal operation |
| 1 | 0 | Loop mode with transmitter output internally connected to receiver input |
| 1 | 1 | Single-wire mode with TXD pin connected to receiver input |

### 20.3.2.3 SCI Alternative Status Register 1 (SCIASR1)

Module Base + 0x0000

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | RXEDGIF | 0 | 0 | 0 | 0 | BERRV | BERRIF | BKDIF |
| W | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented or Reserved

**Figure 20-6. SCI Alternative Status Register 1 (SCIASR1)**

Read: Anytime, if AMAP = 1

Write: Anytime, if AMAP = 1

**Table 20-6. SCIASR1 Field Descriptions**

| Field | Description |
|---|---|
| 7<br>RXEDGIF | **Receive Input Active Edge Interrupt Flag** — RXEDGIF is asserted, if an active edge (falling if RXPOL = 0, rising if RXPOL = 1) on the RXD input occurs. RXEDGIF bit is cleared by writing a “1” to it.<br>0 No active receive on the receive input has occurred<br>1 An active edge on the receive input has occurred |
| 2<br>BERRV | **Bit Error Value** — BERRV reflects the state of the RXD input when the bit error detect circuitry is enabled and a mismatch to the expected value happened. The value is only meaningful, if BERRIF = 1.<br>0 A low input was sampled, when a high was expected<br>1 A high input reassembled, when a low was expected |
| 1<br>BERRIF | **Bit Error Interrupt Flag** — BERRIF is asserted, when the bit error detect circuitry is enabled and if the value sampled at the RXD input does not match the transmitted value. If the BERRIE interrupt enable bit is set an interrupt will be generated. The BERRIF bit is cleared by writing a “1” to it.<br>0 No mismatch detected<br>1 A mismatch has occurred |
| 0<br>BKDIF | **Break Detect Interrupt Flag** — BKDIF is asserted, if the break detect circuitry is enabled and a break signal is received. If the BKDIE interrupt enable bit is set an interrupt will be generated. The BKDIF bit is cleared by writing a “1” to it.<br>0 No break signal was received<br>1 A break signal was received |

### 20.3.2.4 SCI Alternative Control Register 1 (SCIACR1)

Module Base + 0x0001

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | RXEDGIE | 0 | 0 | 0 | 0 | 0 | BERRIE | BKDIE |
| W | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented or Reserved

**Figure 20-7. SCI Alternative Control Register 1 (SCIACR1)**

Read: Anytime, if AMAP = 1

Write: Anytime, if AMAP = 1

**Table 20-7. SCIACR1 Field Descriptions**

| Field | Description |
|---|---|
| 7<br>RSEDGIE | **Receive Input Active Edge Interrupt Enable** — RXEDGIE enables the receive input active edge interrupt flag, RXEDGIF, to generate interrupt requests.<br>0 RXEDGIF interrupt requests disabled<br>1 RXEDGIF interrupt requests enabled |
| 1<br>BERRIE | **Bit Error Interrupt Enable** — BERRIE enables the bit error interrupt flag, BERRIF, to generate interrupt requests.<br>0 BERRIF interrupt requests disabled<br>1 BERRIF interrupt requests enabled |
| 0<br>BKDIE | **Break Detect Interrupt Enable** — BKDIE enables the break detect interrupt flag, BKDIF, to generate interrupt requests.<br>0 BKDIF interrupt requests disabled<br>1 BKDIF interrupt requests enabled |

### 20.3.2.5 SCI Alternative Control Register 2 (SCIACR2)

Module Base + 0x0002

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | 0 | 0 | 0 | BERRM1 | BERRM0 | BKDFE |
| W | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented or Reserved

**Figure 20-8. SCI Alternative Control Register 2 (SCIACR2)**

Read: Anytime, if AMAP = 1

Write: Anytime, if AMAP = 1

**Table 20-8. SCIACR2 Field Descriptions**

| Field | Description |
|---|---|
| 2:1<br>BERRM[1:0] | **Bit Error Mode** — Those two bits determines the functionality of the bit error detect feature. See Table 20-9. |
| 0<br>BKDFE | **Break Detect Feature Enable** — BKDFE enables the break detect circuitry.<br>0 Break detect circuit disabled<br>1 Break detect circuit enabled |

**Table 20-9. Bit Error Mode Coding**

| BERRM1 | BERRM0 | Function |
|---|---|---|
| 0 | 0 | Bit error detect circuit is disabled |
| 0 | 1 | Receive input sampling occurs during the 9th time tick of a transmitted bit (refer to Figure 20-19) |
| 1 | 0 | Receive input sampling occurs during the 13th time tick of a transmitted bit (refer to Figure 20-19) |
| 1 | 1 | Reserved |

### 20.3.2.6 SCI Control Register 2 (SCICR2)

Module Base + 0x0003

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | TIE | TCIE | RIE | ILIE | TE | RE | RWU | SBK |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 20-9. SCI Control Register 2 (SCICR2)**

Read: Anytime

Write: Anytime

**Table 20-10. SCICR2 Field Descriptions**

| Field | Description |
|---|---|
| 7<br>TIE | **Transmitter Interrupt Enable Bit** — TIE enables the transmit data register empty flag, TDRE, to generate interrupt requests.<br>0 TDRE interrupt requests disabled<br>1 TDRE interrupt requests enabled |
| 6<br>TCIE | **Transmission Complete Interrupt Enable Bit** — TCIE enables the transmission complete flag, TC, to generate interrupt requests.<br>0 TC interrupt requests disabled<br>1 TC interrupt requests enabled |
| 5<br>RIE | **Receiver Full Interrupt Enable Bit** — RIE enables the receive data register full flag, RDRF, or the overrun flag, OR, to generate interrupt requests.<br>0 RDRF and OR interrupt requests disabled<br>1 RDRF and OR interrupt requests enabled |
| 4<br>ILIE | **Idle Line Interrupt Enable Bit** — ILIE enables the idle line flag, IDLE, to generate interrupt requests.<br>0 IDLE interrupt requests disabled<br>1 IDLE interrupt requests enabled |
| 3<br>TE | **Transmitter Enable Bit** — TE enables the SCI transmitter and configures the TXD pin as being controlled by the SCI. The TE bit can be used to queue an idle preamble.<br>0 Transmitter disabled<br>1 Transmitter enabled |
| 2<br>RE | **Receiver Enable Bit** — RE enables the SCI receiver.<br>0 Receiver disabled<br>1 Receiver enabled |
| 1<br>RWU | **Receiver Wakeup Bit** — Standby state<br>0 Normal operation.<br>1 RWU enables the wakeup function and inhibits further receiver interrupt requests. Normally, hardware wakes the receiver by automatically clearing RWU. |
| 0<br>SBK | **Send Break Bit** — Toggling SBK sends one break character (10 or 11 logic 0s, respectively 13 or 14 logics 0s if BRK13 is set). Toggling implies clearing the SBK bit before the break character has finished transmitting. As long as SBK is set, the transmitter continues to send complete break characters (10 or 11 bits, respectively 13 or 14 bits).<br>0 No break characters<br>1 Transmit break characters |

### 20.3.2.7 SCI Status Register 1 (SCISR1)

The SCISR1 and SCISR2 registers provides inputs to the MCU for generation of SCI interrupts. Also, these registers can be polled by the MCU to check the status of these bits. The flag-clearing procedures require that the status register be read followed by a read or write to the SCI data register.It is permissible to execute other instructions between the two steps as long as it does not compromise the handling of I/O, but the order of operations is important for flag clearing.

Module Base + 0x0004

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | TDRE | TC | RDRF | IDLE | OR | NF | FE | PF |
| W | | | | | | | | |
| Reset | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented or Reserved

**Figure 20-10. SCI Status Register 1 (SCISR1)**

Read: Anytime

Write: Has no meaning or effect

**Table 20-11. SCISR1 Field Descriptions**

| Field | Description |
|---|---|
| 7<br>TDRE | **Transmit Data Register Empty Flag** — TDRE is set when the transmit shift register receives a byte from the SCI data register. When TDRE is 1, the transmit data register (SCIDRH/L) is empty and can receive a new value to transmit.Clear TDRE by reading SCI status register 1 (SCISR1), with TDRE set and then writing to SCI data register low (SCIDRL).<br>0 No byte transferred to transmit shift register<br>1 Byte transferred to transmit shift register; transmit data register empty |
| 6<br>TC | **Transmit Complete Flag** — TC is set low when there is a transmission in progress or when a preamble or break character is loaded. TC is set high when the TDRE flag is set and no data, preamble, or break character is being transmitted.When TC is set, the TXD pin becomes idle (logic 1). Clear TC by reading SCI status register 1 (SCISR1) with TC set and then writing to SCI data register low (SCIDRL). TC is cleared automatically when data, preamble, or break is queued and ready to be sent. TC is cleared in the event of a simultaneous set and clear of the TC flag (transmission not complete).<br>0 Transmission in progress<br>1 No transmission in progress |
| 5<br>RDRF | **Receive Data Register Full Flag** — RDRF is set when the data in the receive shift register transfers to the SCI data register. Clear RDRF by reading SCI status register 1 (SCISR1) with RDRF set and then reading SCI data register low (SCIDRL).<br>0 Data not available in SCI data register<br>1 Received data available in SCI data register |
| 4<br>IDLE | **Idle Line Flag** — IDLE is set when 10 consecutive logic 1s (if M = 0) or 11 consecutive logic 1s (if M =1) appear on the receiver input. Once the IDLE flag is cleared, a valid frame must again set the RDRF flag before an idle condition can set the IDLE flag.Clear IDLE by reading SCI status register 1 (SCISR1) with IDLE set and then reading SCI data register low (SCIDRL).<br>0 Receiver input is either active now or has never become active since the IDLE flag was last cleared<br>1 Receiver input has become idle<br>**Note:** When the receiver wakeup bit (RWU) is set, an idle line condition does not set the IDLE flag. |

**Table 20-11. SCISR1 Field Descriptions (continued)**

| Field | Description |
|---|---|
| 3<br>OR | **Overrun Flag** — OR is set when software fails to read the SCI data register before the receive shift register receives the next frame. The OR bit is set immediately after the stop bit has been completely received for the second frame. The data in the shift register is lost, but the data already in the SCI data registers is not affected. Clear OR by reading SCI status register 1 (SCISR1) with OR set and then reading SCI data register low (SCIDRL).<br>0 No overrun<br>1 Overrun<br>**Note:** OR flag may read back as set when RDRF flag is clear. This may happen if the following sequence of events occurs:<br>1. After the first frame is received, read status register SCISR1 (returns RDRF set and OR flag clear);<br>2. Receive second frame without reading the first frame in the data register (the second frame is not received and OR flag is set);<br>3. Read data register SCIDRL (returns first frame and clears RDRF flag in the status register);<br>4. Read status register SCISR1 (returns RDRF clear and OR set).<br>Event 3 may be at exactly the same time as event 2 or any time after. When this happens, a dummy SCIDRL read following event 4 will be required to clear the OR flag if further frames are to be received. |
| 2<br>NF | **Noise Flag** — NF is set when the SCI detects noise on the receiver input. NF bit is set during the same cycle as the RDRF flag but does not get set in the case of an overrun. Clear NF by reading SCI status register 1(SCISR1), and then reading SCI data register low (SCIDRL).<br>0 No noise<br>1 Noise |
| 1<br>FE | **Framing Error Flag** — FE is set when a logic 0 is accepted as the stop bit. FE bit is set during the same cycle as the RDRF flag but does not get set in the case of an overrun. FE inhibits further data reception until it is cleared. Clear FE by reading SCI status register 1 (SCISR1) with FE set and then reading the SCI data register low (SCIDRL).<br>0 No framing error<br>1 Framing error |
| 0<br>PF | **Parity Error Flag** — PF is set when the parity enable bit (PE) is set and the parity of the received data does not match the parity type bit (PT). PF bit is set during the same cycle as the RDRF flag but does not get set in the case of an overrun. Clear PF by reading SCI status register 1 (SCISR1), and then reading SCI data register low (SCIDRL).<br>0 No parity error<br>1 Parity error |

### 20.3.2.8 SCI Status Register 2 (SCISR2)

Module Base + 0x0005

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | AMAP | 0 | 0 | TXPOL | RXPOL | BRK13 | TXDIR | RAF |
| W | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented or Reserved

**Figure 20-11. SCI Status Register 2 (SCISR2)**

Read: Anytime

Write: Anytime

**Table 20-12. SCISR2 Field Descriptions**

| Field | Description |
|---|---|
| 7<br>AMAP | **Alternative Map** — This bit controls which registers sharing the same address space are accessible. In the reset condition the SCI behaves as previous versions. Setting AMAP=1 allows the access to another set of control and status registers and hides the baud rate and SCI control Register 1.<br>0 The registers labelled SCIBDH (0x0000),SCIBDL (0x0001), SCICR1 (0x0002) are accessible<br>1 The registers labelled SCIASR1 (0x0000),SCIACR1 (0x0001), SCIACR2 (0x00002) are accessible |
| 4<br>TXPOL | **Transmit Polarity** — This bit control the polarity of the transmitted data. In NRZ format, a one is represented by a mark and a zero is represented by a space for normal polarity, and the opposite for inverted polarity. In IrDA format, a zero is represented by short high pulse in the middle of a bit time remaining idle low for a one for normal polarity, and a zero is represented by short low pulse in the middle of a bit time remaining idle high for a one for inverted polarity.<br>0 Normal polarity<br>1 Inverted polarity |
| 3<br>RXPOL | **Receive Polarity** — This bit control the polarity of the received data. In NRZ format, a one is represented by a mark and a zero is represented by a space for normal polarity, and the opposite for inverted polarity. In IrDA format, a zero is represented by short high pulse in the middle of a bit time remaining idle low for a one for normal polarity, and a zero is represented by short low pulse in the middle of a bit time remaining idle high for a one for inverted polarity.<br>0 Normal polarity<br>1 Inverted polarity |
| 2<br>BRK13 | **Break Transmit Character Length** — This bit determines whether the transmit break character is 10 or 11 bit respectively 13 or 14 bits long. The detection of a framing error is not affected by this bit.<br>0 Break character is 10 or 11 bit long<br>1 Break character is 13 or 14 bit long |
| 1<br>TXDIR | **Transmitter Pin Data Direction in Single-Wire Mode** — This bit determines whether the TXD pin is going to be used as an input or output, in the single-wire mode of operation. This bit is only relevant in the single-wire mode of operation.<br>0 TXD pin to be used as an input in single-wire mode<br>1 TXD pin to be used as an output in single-wire mode |
| 0<br>RAF | **Receiver Active Flag** — RAF is set when the receiver detects a logic 0 during the RT1 time period of the start bit search. RAF is cleared when the receiver detects an idle character.<br>0 No reception in progress<br>1 Reception in progress |

### 20.3.2.9 SCI Data Registers (SCIDRH, SCIDRL)

Module Base + 0x0006

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | R8 | T8 | 0 | 0 | 0 | 0 | 0 | 0 |
| W | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented or Reserved

**Figure 20-12. SCI Data Registers (SCIDRH)**

Module Base + 0x0007

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | R7 | R6 | R5 | R4 | R3 | R2 | R1 | R0 |
| W | T7 | T6 | T5 | T4 | T3 | T2 | T1 | T0 |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 20-13. SCI Data Registers (SCIDRL)**

Read: Anytime; reading accesses SCI receive data register

Write: Anytime; writing accesses SCI transmit data register; writing to R8 has no effect

**Table 20-13. SCIDRH and SCIDRL Field Descriptions**

| Field | Description |
|---|---|
| SCIDRH<br>7<br>R8 | **Received Bit 8** — R8 is the ninth data bit received when the SCI is configured for 9-bit data format (M = 1). |
| SCIDRH<br>6<br>T8 | **Transmit Bit 8** — T8 is the ninth data bit transmitted when the SCI is configured for 9-bit data format (M = 1). |
| SCIDRL<br>7:0<br>R[7:0]<br>T[7:0] | **R7:R0** — Received bits seven through zero for 9-bit or 8-bit data formats<br>**T7:T0** — Transmit bits seven through zero for 9-bit or 8-bit formats |

**NOTE**

If the value of T8 is the same as in the previous transmission, T8 does not have to be rewritten.The same value is transmitted until T8 is rewritten

In 8-bit data format, only SCI data register low (SCIDRL) needs to be accessed.

When transmitting in 9-bit data format and using 8-bit write instructions, write first to SCI data register high (SCIDRH), then SCIDRL.

## 20.4 Functional Description

This section provides a complete functional description of the SCI block, detailing the operation of the design from the end user perspective in a number of subsections.

Figure 20-14 shows the structure of the SCI module. The SCI allows full duplex, asynchronous, serial communication between the CPU and remote devices, including other CPUs. The SCI transmitter and receiver operate independently, although they use the same baud rate generator. The CPU monitors the status of the SCI, writes the data to be transmitted, and processes received data.

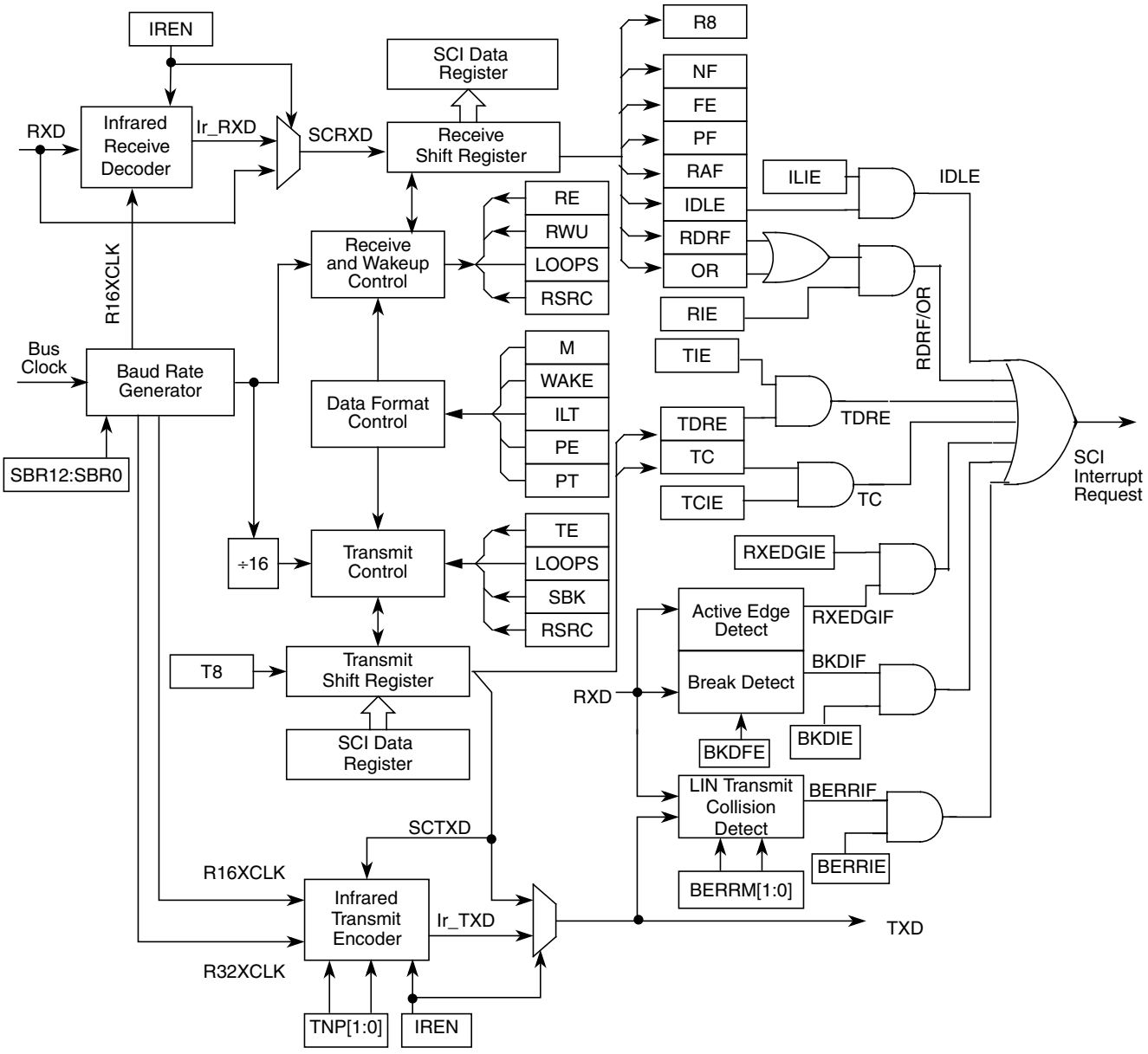



**Figure 20-14. Detailed SCI Block Diagram**

### 20.4.1 Infrared Interface Submodule

This module provides the capability of transmitting narrow pulses to an IR LED and receiving narrow pulses and transforming them to serial bits, which are sent to the SCI. The IrDA physical layer specification defines a half-duplex infrared communication link for exchange data. The full standard includes data rates up to 16 Mbits/s. This design covers only data rates between 2.4 Kbits/s and 115.2 Kbits/s.

The infrared submodule consists of two major blocks: the transmit encoder and the receive decoder. The SCI transmits serial bits of data which are encoded by the infrared submodule to transmit a narrow pulse for every zero bit. No pulse is transmitted for every one bit. When receiving data, the IR pulses should be detected using an IR photo diode and transformed to CMOS levels by the IR receive decoder (external from the MCU). The narrow pulses are then stretched by the infrared submodule to get back to a serial bit stream to be received by the SCI.The polarity of transmitted pulses and expected receive pulses can be inverted so that a direct connection can be made to external IrDA transceiver modules that use active low pulses.

The infrared submodule receives its clock sources from the SCI. One of these two clocks are selected in the infrared submodule in order to generate either 3/16, 1/16, 1/32 or 1/4 narrow pulses during transmission. The infrared block receives two clock sources from the SCI, R16XCLK and R32XCLK, which are configured to generate the narrow pulse width during transmission. The R16XCLK and R32XCLK are internal clocks with frequencies 16 and 32 times the baud rate respectively. Both R16XCLK and R32XCLK clocks are used for transmitting data. The receive decoder uses only the R16XCLK clock.

#### 20.4.1.1 Infrared Transmit Encoder

The infrared transmit encoder converts serial bits of data from transmit shift register to the TXD pin. A narrow pulse is transmitted for a zero bit and no pulse for a one bit. The narrow pulse is sent in the middle of the bit with a duration of 1/32, 1/16, 3/16 or 1/4 of a bit time. A narrow high pulse is transmitted for a zero bit when TXPOL is cleared, while a narrow low pulse is transmitted for a zero bit when TXPOL is set.

#### 20.4.1.2 Infrared Receive Decoder

The infrared receive block converts data from the RXD pin to the receive shift register. A narrow pulse is expected for each zero received and no pulse is expected for each one received. A narrow high pulse is expected for a zero bit when RXPOL is cleared, while a narrow low pulse is expected for a zero bit when RXPOL is set. This receive decoder meets the edge jitter requirement as defined by the IrDA serial infrared physical layer specification.

### 20.4.2 LIN Support

This module provides some basic support for the LIN protocol. At first this is a break detect circuitry making it easier for the LIN software to distinguish a break character from an incoming data stream. As a further addition is supports a collision detection at the bit level as well as cancelling pending transmissions.

### 20.4.3 Data Format

The SCI uses the standard NRZ mark/space data format. When Infrared is enabled, the SCI uses RZI data format where zeroes are represented by light pulses and ones remain low. See Figure 20-15 below.

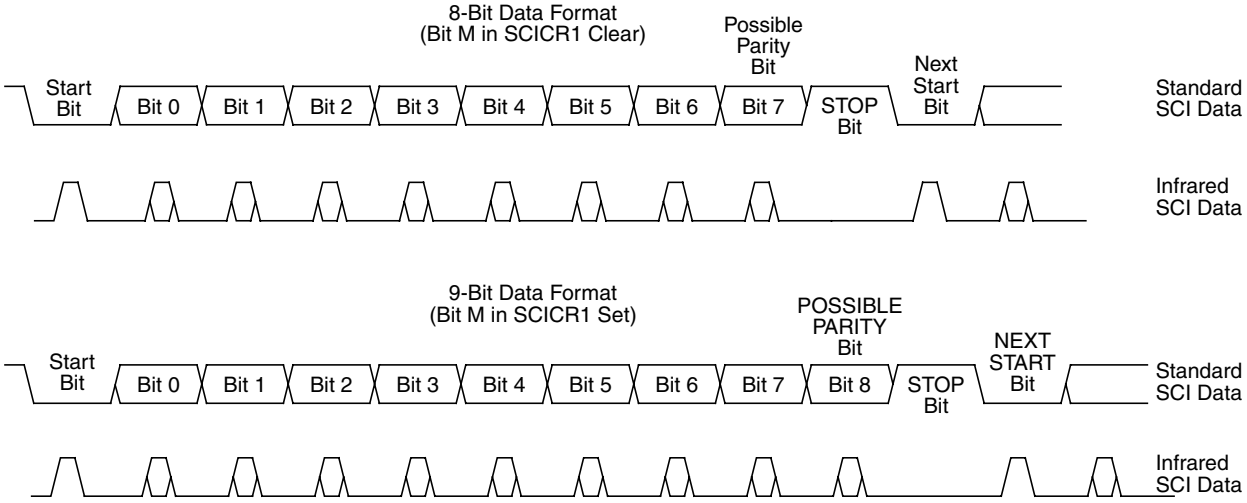



**Figure 20-15. SCI Data Formats**

Each data character is contained in a frame that includes a start bit, eight or nine data bits, and a stop bit. Clearing the M bit in SCI control register 1 configures the SCI for 8-bit data characters. A frame with eight data bits has a total of 10 bits. Setting the M bit configures the SCI for nine-bit data characters. A frame with nine data bits has a total of 11 bits.

**Table 20-14. Example of 8-Bit Data Formats**

| Start Bit | Data Bits | Address Bits | Parity Bits | Stop Bit |
|---|---|---|---|---|
| 1 | 8 | 0 | 0 | 1 |
| 1 | 7 | 0 | 1 | 1 |
| 1 | 7 | 1[1] | 0 | 1 |

1. The address bit identifies the frame as an address character. See Section 20.4.6.6, “Receiver Wakeup”.

When the SCI is configured for 9-bit data characters, the ninth data bit is the T8 bit in SCI data register high (SCIDRH). It remains unchanged after transmission and can be used repeatedly without rewriting it. A frame with nine data bits has a total of 11 bits.

**Table 20-15. Example of 9-Bit Data Formats**

| Start Bit | Data Bits | Address Bits | Parity Bits | Stop Bit |
|---|---|---|---|---|
| 1 | 9 | 0 | 0 | 1 |
| 1 | 8 | 0 | 1 | 1 |
| 1 | 8 | 1[1] | 0 | 1 |

1. The address bit identifies the frame as an address character. See Section 20.4.6.6, “Receiver Wakeup”.

## 20.4.4 Baud Rate Generation

A 13-bit modulus counter in the baud rate generator derives the baud rate for both the receiver and the transmitter. The value from 0 to 8191 written to the SBR12:SBR0 bits determines the bus clock divisor. The SBR bits are in the SCI baud rate registers (SCIBDH and SCIBDL). The baud rate clock is synchronized with the bus clock and drives the receiver. The baud rate clock divided by 16 drives the transmitter. The receiver has an acquisition rate of 16 samples per bit time.

Baud rate generation is subject to one source of error:

- Integer division of the bus clock may not give the exact target frequency.

Table 20-16 lists some examples of achieving target baud rates with a bus clock frequency of 25 MHz.

When IREN = 0 then,

SCI baud rate = SCI bus clock / (16 * SCIBR[12:0])

**Table 20-16. Baud Rates (Example: Bus Clock = 25 MHz)**

| Bits SBR[12:0] | Receiver Clock (Hz) | Transmitter Clock (Hz) | Target Baud Rate | Error (%) |
|---|---|---|---|---|
| 41 | 609,756.1 | 38,109.8 | 38,400 | .76 |
| 81 | 308,642.0 | 19,290.1 | 19,200 | .47 |
| 163 | 153,374.2 | 9585.9 | 9,600 | .16 |
| 326 | 76,687.1 | 4792.9 | 4,800 | .15 |
| 651 | 38,402.5 | 2400.2 | 2,400 | .01 |
| 1302 | 19,201.2 | 1200.1 | 1,200 | .01 |
| 2604 | 9600.6 | 600.0 | 600 | .00 |
| 5208 | 4800.0 | 300.0 | 300 | .00 |

## 20.4.5 Transmitter

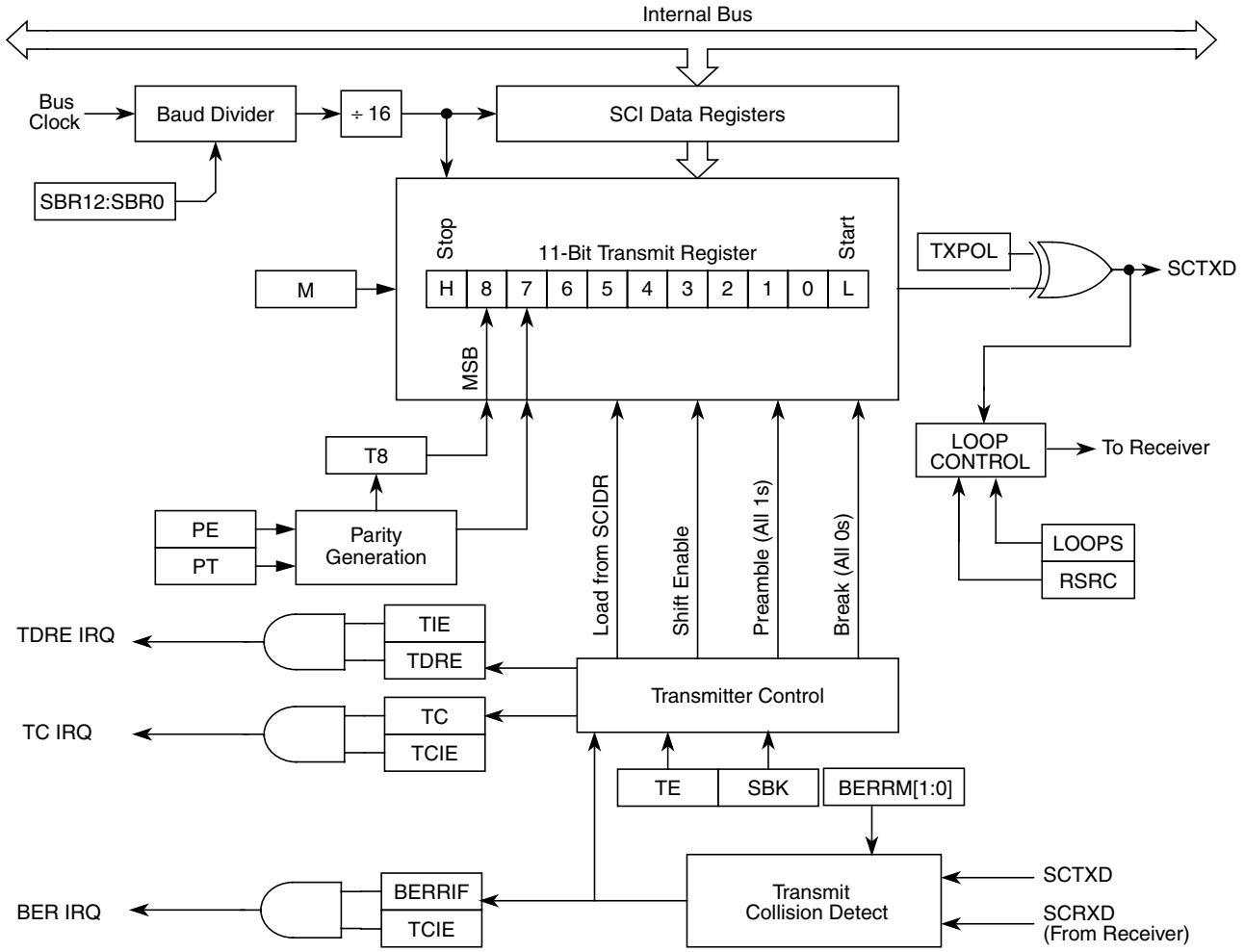



**Figure 20-16. Transmitter Block Diagram**

### 20.4.5.1 Transmitter Character Length

The SCI transmitter can accommodate either 8-bit or 9-bit data characters. The state of the M bit in SCI control register 1 (SCICR1) determines the length of data characters. When transmitting 9-bit data, bit T8 in SCI data register high (SCIDRH) is the ninth bit (bit 8).

### 20.4.5.2 Character Transmission

To transmit data, the MCU writes the data bits to the SCI data registers (SCIDRH/SCIDRL), which in turn are transferred to the transmitter shift register. The transmit shift register then shifts a frame out through the TXD pin, after it has prefaced them with a start bit and appended them with a stop bit. The SCI data registers (SCIDRH and SCIDRL) are the write-only buffers between the internal data bus and the transmit shift register.

The SCI also sets a flag, the transmit data register empty flag (TDRE), every time it transfers data from the buffer (SCIDRH/L) to the transmitter shift register.The transmit driver routine may respond to this flag by writing another byte to the Transmitter buffer (SCIDRH/SCIDRL), while the shift register is still shifting out the first byte.

To initiate an SCI transmission:

1. Configure the SCI:
   a) Select a baud rate. Write this value to the SCI baud registers (SCIBDH/L) to begin the baud rate generator. Remember that the baud rate generator is disabled when the baud rate is zero. Writing to the SCIBDH has no effect without also writing to SCIBDL.
   b) Write to SCICR1 to configure word length, parity, and other configuration bits (LOOPS,RSRC,M,WAKE,ILT,PE,PT).
   c) Enable the transmitter, interrupts, receive, and wake up as required, by writing to the SCICR2 register bits (TIE,TCIE,RIE,ILIE,TE,RE,RWU,SBK). A preamble or idle character will now be shifted out of the transmitter shift register.
2. Transmit Procedure for each byte:
   a) Poll the TDRE flag by reading the SCISR1 or responding to the TDRE interrupt. Keep in mind that the TDRE bit resets to one.
   b) If the TDRE flag is set, write the data to be transmitted to SCIDRH/L, where the ninth bit is written to the T8 bit in SCIDRH if the SCI is in 9-bit data format. A new transmission will not result until the TDRE flag has been cleared.
3. Repeat step 2 for each subsequent transmission.

**NOTE**

The TDRE flag is set when the shift register is loaded with the next data to be transmitted from SCIDRH/L, which happens, generally speaking, a little over half-way through the stop bit of the previous frame. Specifically, this transfer occurs 9/16ths of a bit time AFTER the start of the stop bit of the previous frame.

Writing the TE bit from 0 to a 1 automatically loads the transmit shift register with a preamble of 10 logic 1s (if M = 0) or 11 logic 1s (if M = 1). After the preamble shifts out, control logic transfers the data from the SCI data register into the transmit shift register. A logic 0 start bit automatically goes into the least significant bit position of the transmit shift register. A logic 1 stop bit goes into the most significant bit position.

Hardware supports odd or even parity. When parity is enabled, the most significant bit (MSB) of the data character is the parity bit.

The transmit data register empty flag, TDRE, in SCI status register 1 (SCISR1) becomes set when the SCI data register transfers a byte to the transmit shift register. The TDRE flag indicates that the SCI data register can accept new data from the internal data bus. If the transmit interrupt enable bit, TIE, in SCI control register 2 (SCICR2) is also set, the TDRE flag generates a transmitter interrupt request.

When the transmit shift register is not transmitting a frame, the TXD pin goes to the idle condition, logic 1. If at any time software clears the TE bit in SCI control register 2 (SCICR2), the transmitter enable signal goes low and the transmit signal goes idle.

If software clears TE while a transmission is in progress (TC = 0), the frame in the transmit shift register continues to shift out. To avoid accidentally cutting off the last frame in a message, always wait for TDRE to go high after the last frame before clearing TE.

To separate messages with preambles with minimum idle line time, use this sequence between messages:

1. Write the last byte of the first message to SCIDRH/L.
2. Wait for the TDRE flag to go high, indicating the transfer of the last frame to the transmit shift register.
3. Queue a preamble by clearing and then setting the TE bit.
4. Write the first byte of the second message to SCIDRH/L.

#### 20.4.5.3 Break Characters

Writing a logic 1 to the send break bit, SBK, in SCI control register 2 (SCICR2) loads the transmit shift register with a break character. A break character contains all logic 0s and has no start, stop, or parity bit. Break character length depends on the M bit in SCI control register 1 (SCICR1). As long as SBK is at logic 1, transmitter logic continuously loads break characters into the transmit shift register. After software clears the SBK bit, the shift register finishes transmitting the last break character and then transmits at least one logic 1. The automatic logic 1 at the end of a break character guarantees the recognition of the start bit of the next frame.

The SCI recognizes a break character when there are 10 or 11(M = 0 or M = 1) consecutive zero received. Depending if the break detect feature is enabled or not receiving a break character has these effects on SCI registers.

If the break detect feature is disabled (BKDFE = 0):

- Sets the framing error flag, FE
- Sets the receive data register full flag, RDRF
- Clears the SCI data registers (SCIDRH/L)
- May set the overrun flag, OR, noise flag, NF, parity error flag, PE, or the receiver active flag, RAF (see 3.4.4 and 3.4.5 SCI Status Register 1 and 2)

If the break detect feature is enabled (BKDFE = 1) there are two scenarios[1]

The break is detected right from a start bit or is detected during a byte reception.

- Sets the break detect interrupt flag, BKDIF
- Does not change the data register full flag, RDRF or overrun flag OR
- Does not change the framing error flag FE, parity error flag PE.
- Does not clear the SCI data registers (SCIDRH/L)
- May set noise flag NF, or receiver active flag RAF.

1. A Break character in this context are either 10 or 11 consecutive zero received bits

Figure 20-17 shows two cases of break detect. In trace RXD_1 the break symbol starts with the start bit, while in RXD_2 the break starts in the middle of a transmission. If BRKDFE = 1, in RXD_1 case there will be no byte transferred to the receive buffer and the RDRF flag will not be modified. Also no framing error or parity error will be flagged from this transfer. In RXD_2 case, however the break signal starts later during the transmission. At the expected stop bit position the byte received so far will be transferred to the receive buffer, the receive data register full flag will be set, a framing error and if enabled and appropriate a parity error will be set. Once the break is detected the BRKDIF flag will be set.

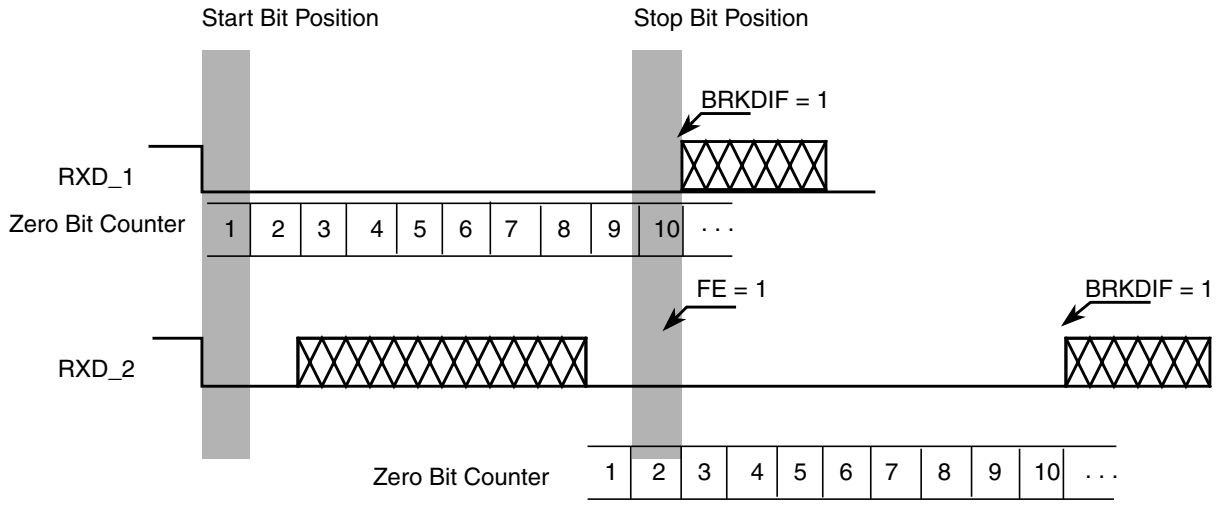



**Figure 20-17. Break Detection if BRKDFE = 1 (M = 0)**

### 20.4.5.4 Idle Characters

An idle character (or preamble) contains all logic 1s and has no start, stop, or parity bit. Idle character length depends on the M bit in SCI control register 1 (SCICR1). The preamble is a synchronizing idle character that begins the first transmission initiated after writing the TE bit from 0 to 1.

If the TE bit is cleared during a transmission, the TXD pin becomes idle after completion of the transmission in progress. Clearing and then setting the TE bit during a transmission queues an idle character to be sent after the frame currently being transmitted.

**NOTE**

When queueing an idle character, return the TE bit to logic 1 before the stop bit of the current frame shifts out through the TXD pin. Setting TE after the stop bit appears on TXD causes data previously written to the SCI data register to be lost. Toggle the TE bit for a queued idle character while the TDRE flag is set and immediately before writing the next byte to the SCI data register.

If the TE bit is clear and the transmission is complete, the SCI is not the master of the TXD pin

### 20.4.5.5 LIN Transmit Collision Detection

This module allows to check for collisions on the LIN bus.

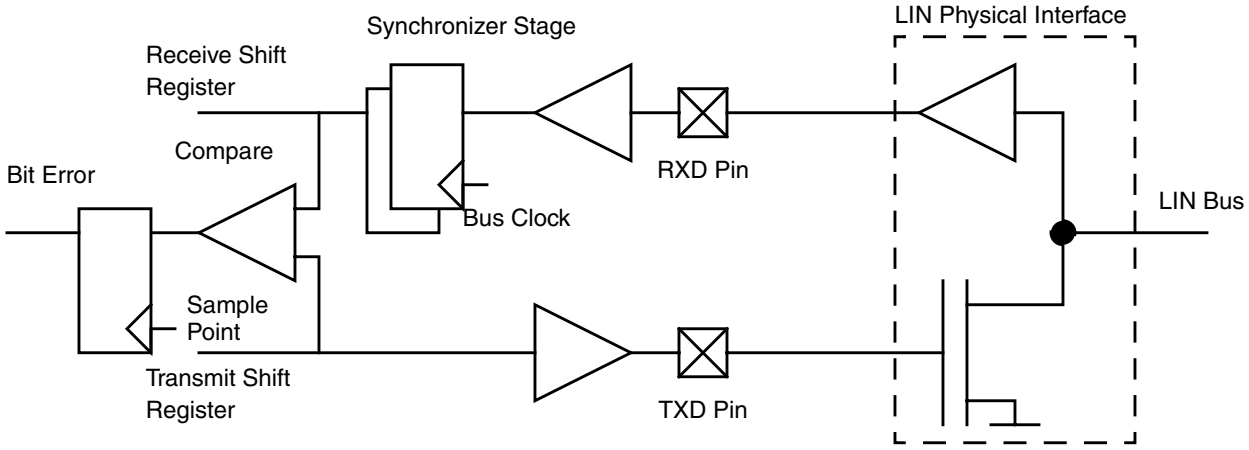



**Figure 20-18. Collision Detect Principle**

If the bit error circuit is enabled (BERRM[1:0] = 0:1 or = 1:0]), the error detect circuit will compare the transmitted and the received data stream at a point in time and flag any mismatch. The timing checks run when transmitter is active (not idle). As soon as a mismatch between the transmitted data and the received data is detected the following happens:

- The next bit transmitted will have a high level (TXPOL = 0) or low level (TXPOL = 1)
- The transmission is aborted and the byte in transmit buffer is discarded.
- the transmit data register empty and the transmission complete flag will be set
- The bit error interrupt flag, BERRIF, will be set.
- No further transmissions will take place until the BERRIF is cleared.

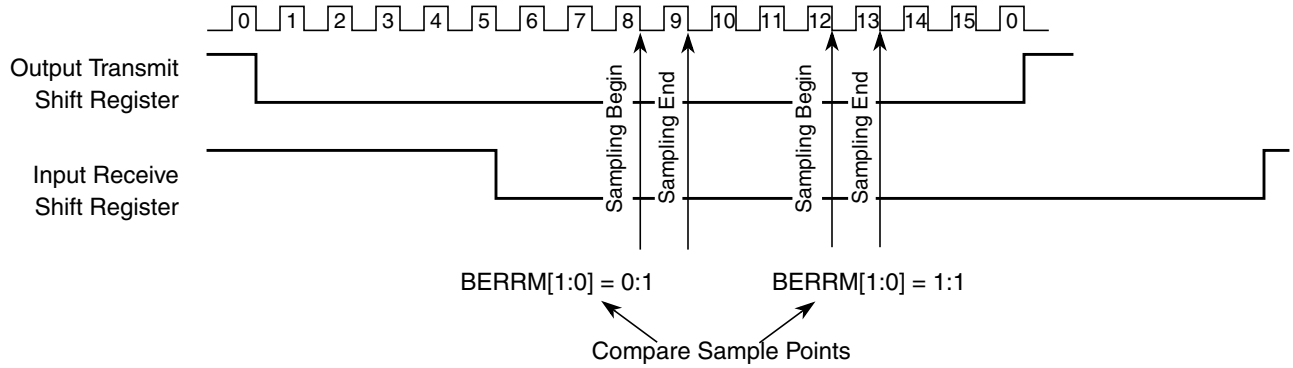



**Figure 20-19. Timing Diagram Bit Error Detection**

If the bit error detect feature is disabled, the bit error interrupt flag is cleared.

**NOTE**

The RXPOL and TXPOL bit should be set the same when transmission collision detect feature is enabled, otherwise the bit error interrupt flag may be set incorrectly.

## 20.4.6 Receiver

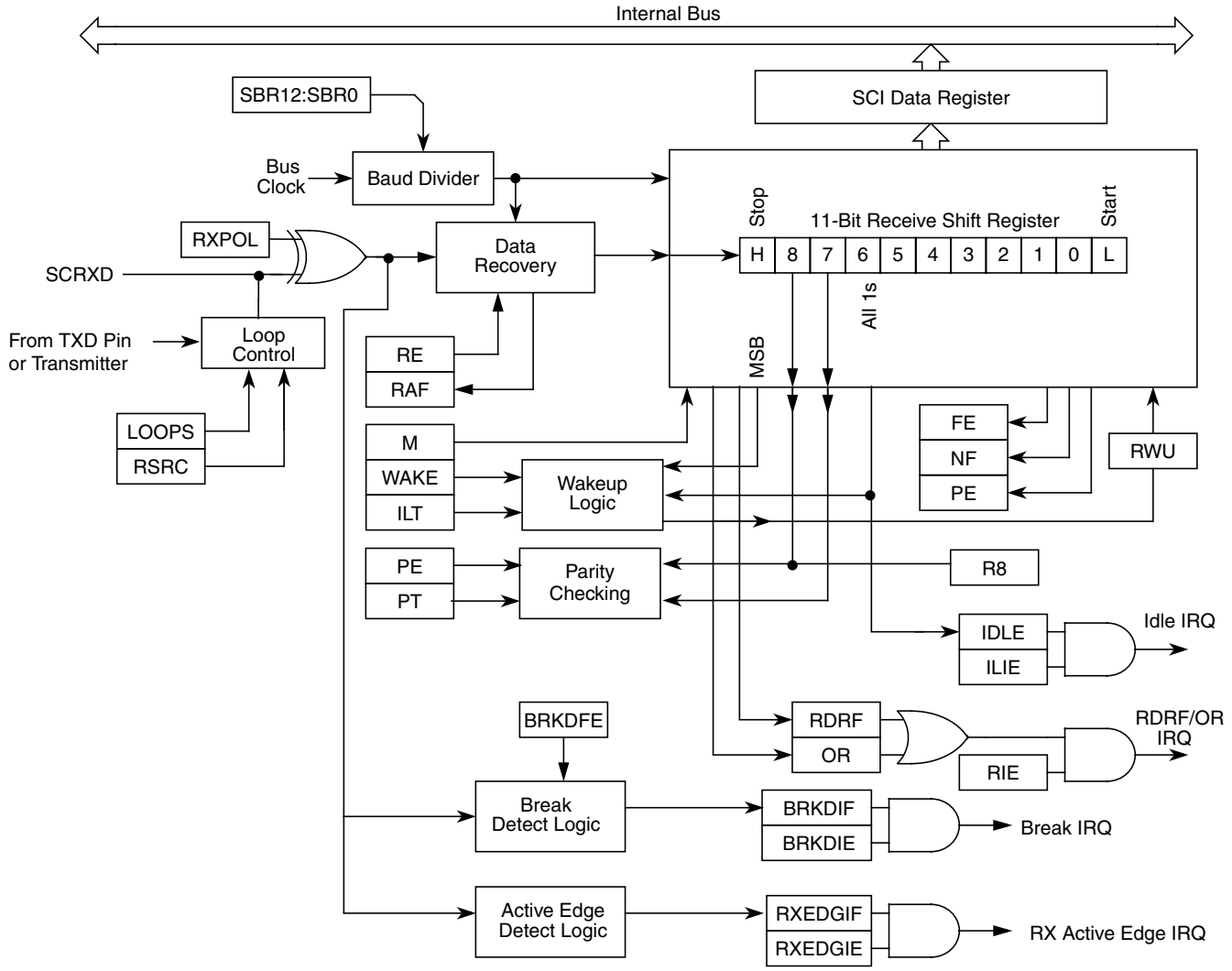



**Figure 20-20. SCI Receiver Block Diagram**

### 20.4.6.1 Receiver Character Length

The SCI receiver can accommodate either 8-bit or 9-bit data characters. The state of the M bit in SCI control register 1 (SCICR1) determines the length of data characters. When receiving 9-bit data, bit R8 in SCI data register high (SCIDRH) is the ninth bit (bit 8).

### 20.4.6.2 Character Reception

During an SCI reception, the receive shift register shifts a frame in from the RXD pin. The SCI data register is the read-only buffer between the internal data bus and the receive shift register.

After a complete frame shifts into the receive shift register, the data portion of the frame transfers to the SCI data register. The receive data register full flag, RDRF, in SCI status register 1 (SCISR1) becomes set,

indicating that the received byte can be read. If the receive interrupt enable bit, RIE, in SCI control register 2 (SCICR2) is also set, the RDRF flag generates an RDRF interrupt request.

### 20.4.6.3 Data Sampling

The RT clock rate. The RT clock is an internal signal with a frequency 16 times the baud rate. To adjust for baud rate mismatch, the RT clock (see Figure 20-21) is re-synchronized:

- After every start bit
- After the receiver detects a data bit change from logic 1 to logic 0 (after the majority of data bit samples at RT8, RT9, and RT10 returns a valid logic 1 and the majority of the next RT8, RT9, and RT10 samples returns a valid logic 0)

To locate the start bit, data recovery logic does an asynchronous search for a logic 0 preceded by three logic 1s.When the falling edge of a possible start bit occurs, the RT clock begins to count to 16.

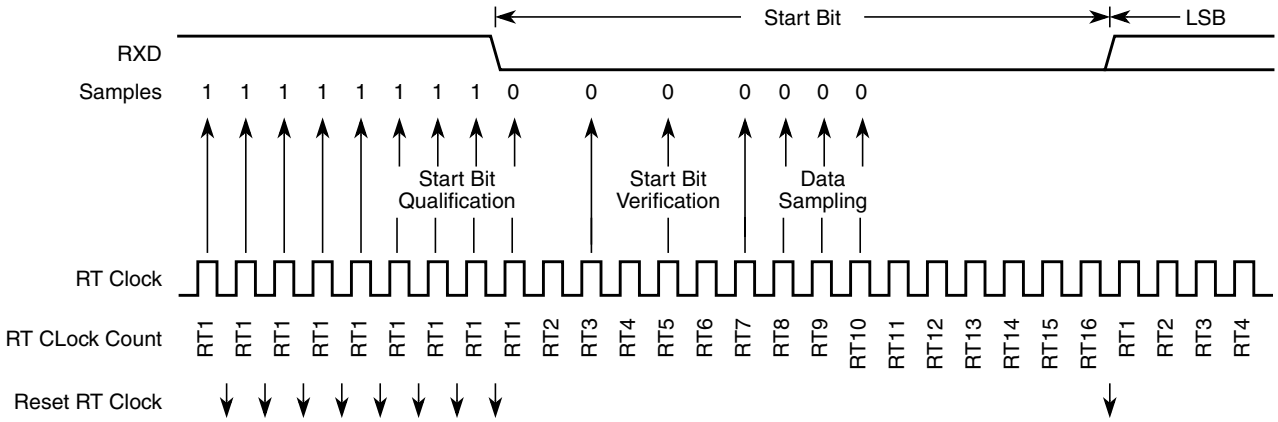



**Figure 20-21. Receiver Data Sampling**

To verify the start bit and to detect noise, data recovery logic takes samples at RT3, RT5, and RT7. Figure 20-17 summarizes the results of the start bit verification samples.

**Table 20-17. Start Bit Verification**

| RT3, RT5, and RT7 Samples | Start Bit Verification | Noise Flag |
|---|---|---|
| 000 | Yes | 0 |
| 001 | Yes | 1 |
| 010 | Yes | 1 |
| 011 | No | 0 |
| 100 | Yes | 1 |
| 101 | No | 0 |
| 110 | No | 0 |
| 111 | No | 0 |

If start bit verification is not successful, the RT clock is reset and a new search for a start bit begins.

To determine the value of a data bit and to detect noise, recovery logic takes samples at RT8, RT9, and RT10. Table 20-18 summarizes the results of the data bit samples.

**Table 20-18. Data Bit Recovery**

| RT8, RT9, and RT10 Samples | Data Bit Determination | Noise Flag |
|---|---|---|
| 000 | 0 | 0 |
| 001 | 0 | 1 |
| 010 | 0 | 1 |
| 011 | 1 | 1 |
| 100 | 0 | 1 |
| 101 | 1 | 1 |
| 110 | 1 | 1 |
| 111 | 1 | 0 |

**NOTE**

The RT8, RT9, and RT10 samples do not affect start bit verification. If any or all of the RT8, RT9, and RT10 start bit samples are logic 1s following a successful start bit verification, the noise flag (NF) is set and the receiver assumes that the bit is a start bit (logic 0).

To verify a stop bit and to detect noise, recovery logic takes samples at RT8, RT9, and RT10. Table 20-19 summarizes the results of the stop bit samples.

**Table 20-19. Stop Bit Recovery**

| RT8, RT9, and RT10 Samples | Framing Error Flag | Noise Flag |
|---|---|---|
| 000 | 1 | 0 |
| 001 | 1 | 1 |
| 010 | 1 | 1 |
| 011 | 0 | 1 |
| 100 | 1 | 1 |
| 101 | 0 | 1 |
| 110 | 0 | 1 |
| 111 | 0 | 0 |

In Figure 20-22 the verification samples RT3 and RT5 determine that the first low detected was noise and not the beginning of a start bit. The RT clock is reset and the start bit search begins again. The noise flag is not set because the noise occurred before the start bit was found.

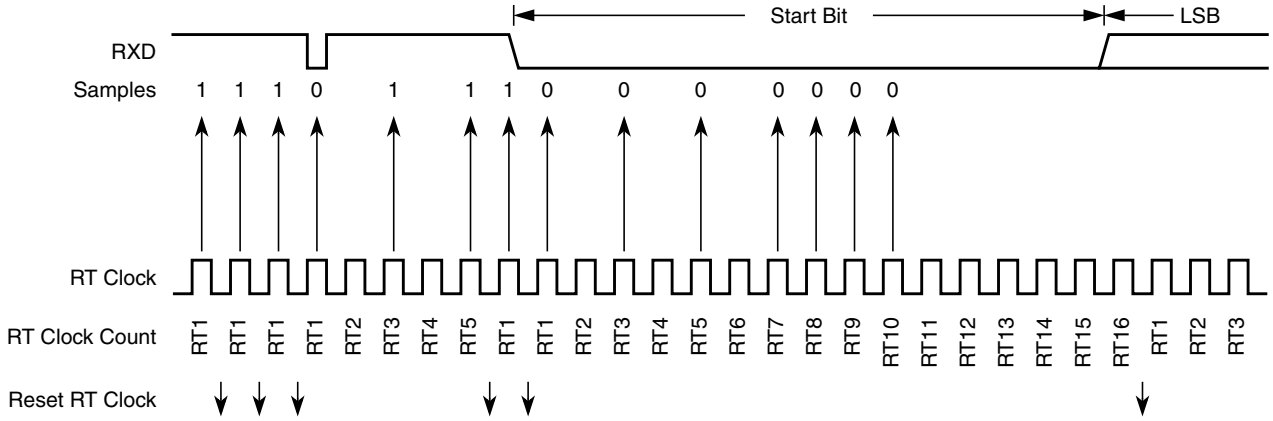



**Figure 20-22. Start Bit Search Example 1**

In Figure 20-23, verification sample at RT3 is high. The RT3 sample sets the noise flag. Although the perceived bit time is misaligned, the data samples RT8, RT9, and RT10 are within the bit time and data recovery is successful.

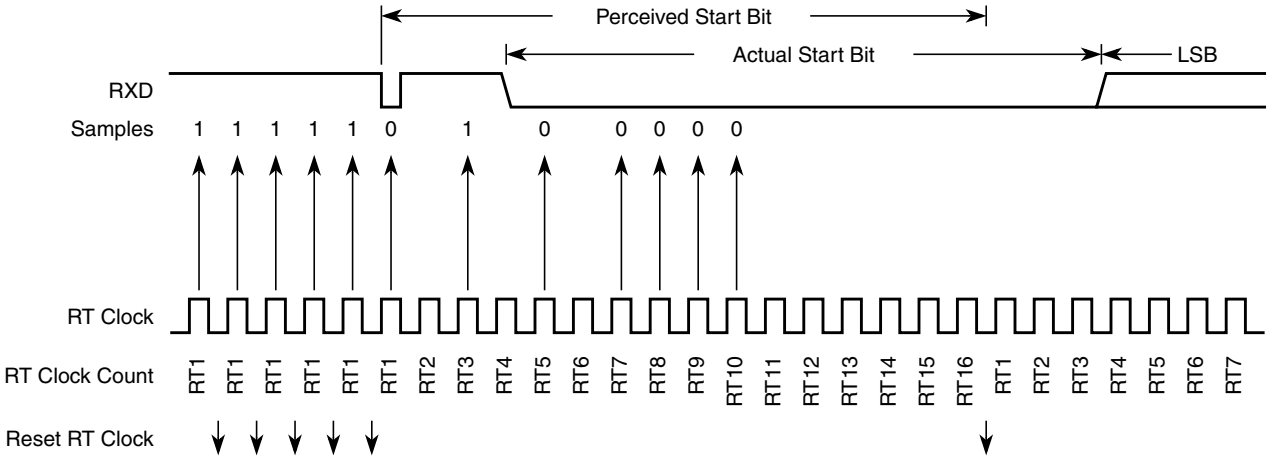



**Figure 20-23. Start Bit Search Example 2**

In Figure 20-24, a large burst of noise is perceived as the beginning of a start bit, although the test sample at RT5 is high. The RT5 sample sets the noise flag. Although this is a worst-case misalignment of perceived bit time, the data samples RT8, RT9, and RT10 are within the bit time and data recovery is successful.

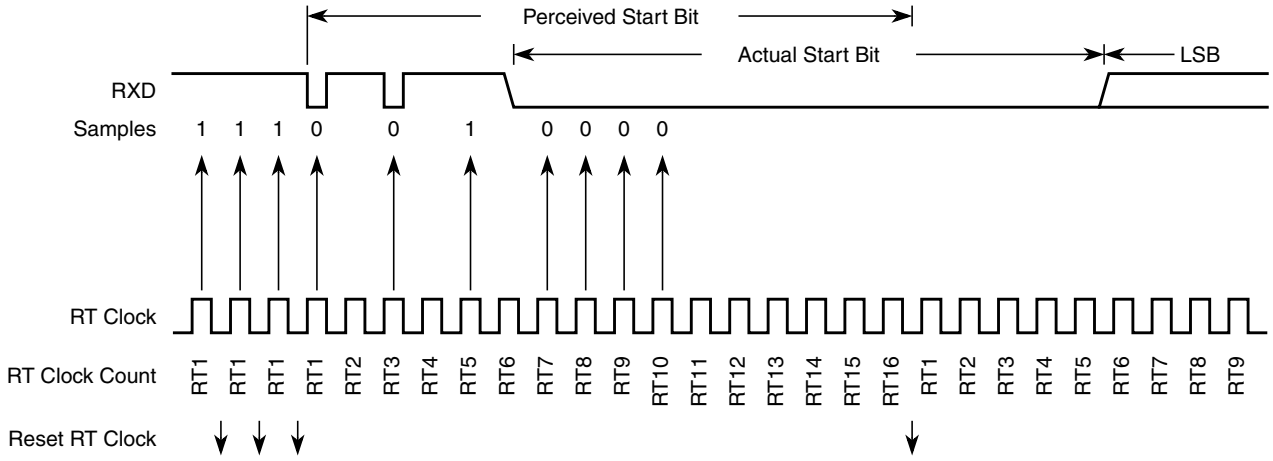



**Figure 20-24. Start Bit Search Example 3**

Figure 20-25 shows the effect of noise early in the start bit time. Although this noise does not affect proper synchronization with the start bit time, it does set the noise flag.

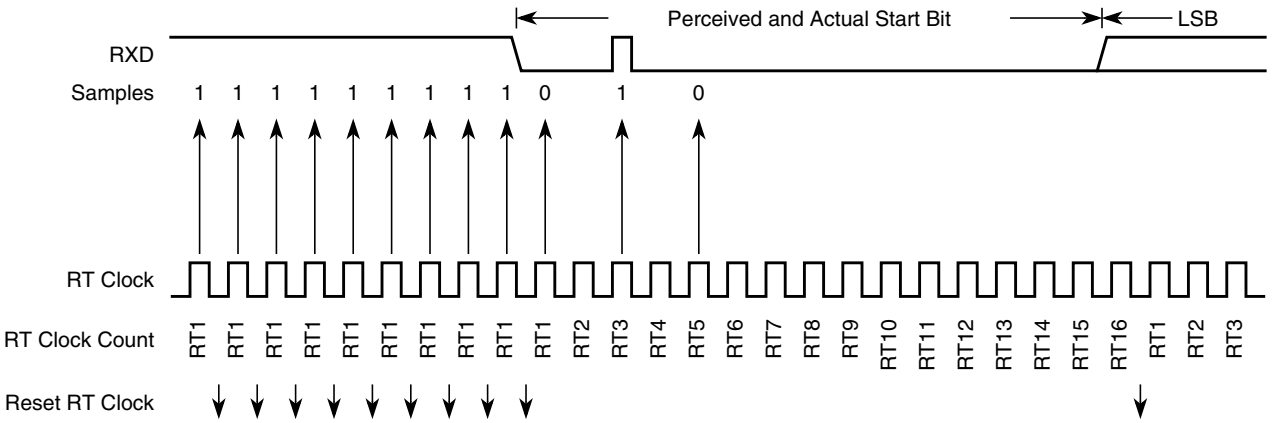



**Figure 20-25. Start Bit Search Example 4**

Figure 20-26 shows a burst of noise near the beginning of the start bit that resets the RT clock. The sample after the reset is low but is not preceded by three high samples that would qualify as a falling edge. Depending on the timing of the start bit search and on the data, the frame may be missed entirely or it may set the framing error flag.

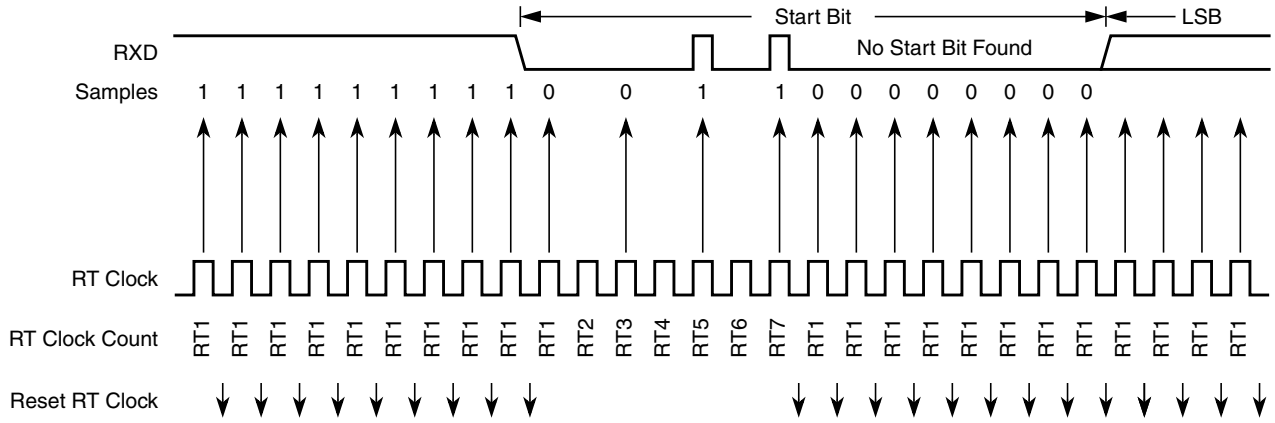



**Figure 20-26. Start Bit Search Example 5**

In Figure 20-27, a noise burst makes the majority of data samples RT8, RT9, and RT10 high. This sets the noise flag but does not reset the RT clock. In start bits only, the RT8, RT9, and RT10 data samples are ignored.

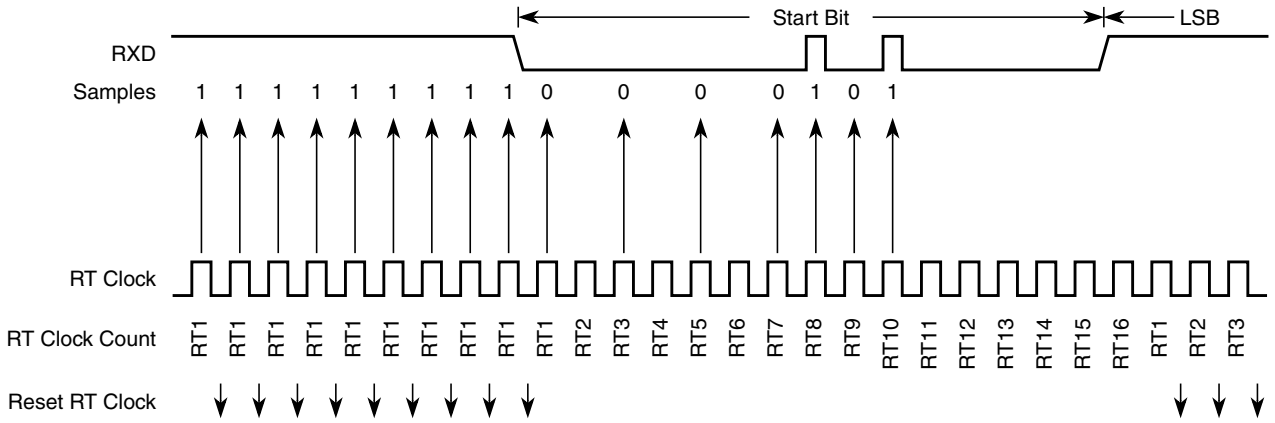



**Figure 20-27. Start Bit Search Example 6**

### 20.4.6.4 Framing Errors

If the data recovery logic does not detect a logic 1 where the stop bit should be in an incoming frame, it sets the framing error flag, FE, in SCI status register 1 (SCISR1). A break character also sets the FE flag because a break character has no stop bit. The FE flag is set at the same time that the RDRF flag is set.

### 20.4.6.5 Baud Rate Tolerance

A transmitting device may be operating at a baud rate below or above the receiver baud rate. Accumulated bit time misalignment can cause one of the three stop bit data samples (RT8, RT9, and RT10) to fall outside the actual stop bit. A noise error will occur if the RT8, RT9, and RT10 samples are not all the same logical values. A framing error will occur if the receiver clock is misaligned in such a way that the majority of the RT8, RT9, and RT10 stop bit samples are a logic zero.

As the receiver samples an incoming frame, it re-synchronizes the RT clock on any valid falling edge within the frame. Re synchronization within frames will correct a misalignment between transmitter bit times and receiver bit times.

#### 20.4.6.5.1 Slow Data Tolerance

Figure 20-28 shows how much a slow received frame can be misaligned without causing a noise error or a framing error. The slow stop bit begins at RT8 instead of RT1 but arrives in time for the stop bit data samples at RT8, RT9, and RT10.

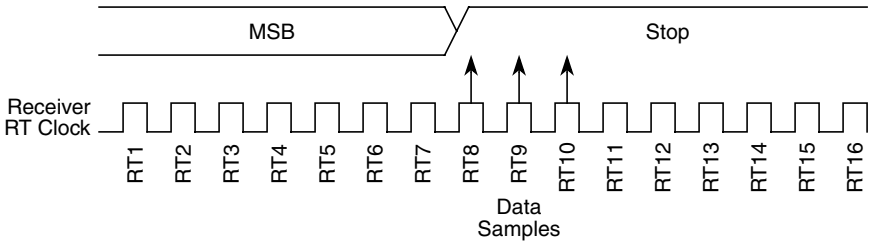



**Figure 20-28. Slow Data**

Let's take RTr as receiver RT clock and RTt as transmitter RT clock.

For an 8-bit data character, it takes the receiver 9 bit times x 16 RTr cycles +7 RTr cycles = 151 RTr cycles to start data sampling of the stop bit.

With the misaligned character shown in Figure 20-28, the receiver counts 151 RTr cycles at the point when the count of the transmitting device is 9 bit times x 16 RTt cycles = 144 RTt cycles.

The maximum percent difference between the receiver count and the transmitter count of a slow 8-bit data character with no errors is:

((151 – 144) / 151) x 100 = 4.63%

For a 9-bit data character, it takes the receiver 10 bit times x 16 RTr cycles + 7 RTr cycles = 167 RTr cycles to start data sampling of the stop bit.

With the misaligned character shown in Figure 20-28, the receiver counts 167 RTr cycles at the point when the count of the transmitting device is 10 bit times x 16 RTt cycles = 160 RTt cycles.

The maximum percent difference between the receiver count and the transmitter count of a slow 9-bit character with no errors is:

((167 – 160) / 167) X 100 = 4.19%

#### 20.4.6.5.2 Fast Data Tolerance

Figure 20-29 shows how much a fast received frame can be misaligned. The fast stop bit ends at RT10 instead of RT16 but is still sampled at RT8, RT9, and RT10.

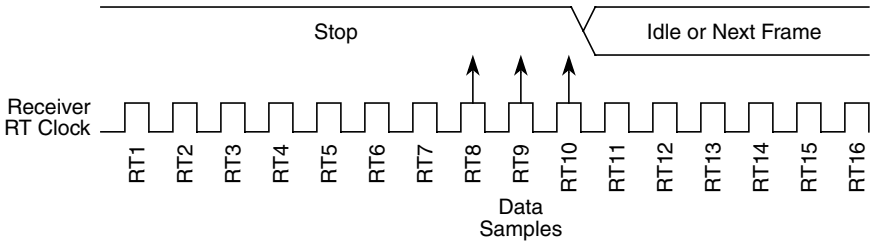



**Figure 20-29. Fast Data**

For an 8-bit data character, it takes the receiver 9 bit times x 16 RTr cycles + 10 RTr cycles = 154 RTr cycles to finish data sampling of the stop bit.

With the misaligned character shown in Figure 20-29, the receiver counts 154 RTr cycles at the point when the count of the transmitting device is 10 bit times x 16 RTt cycles = 160 RTt cycles.

The maximum percent difference between the receiver count and the transmitter count of a fast 8-bit character with no errors is:

((160 – 154) / 160) x 100 = 3.75%

For a 9-bit data character, it takes the receiver 10 bit times x 16 RTr cycles + 10 RTr cycles = 170 RTr cycles to finish data sampling of the stop bit.

With the misaligned character shown in Figure 20-29, the receiver counts 170 RTr cycles at the point when the count of the transmitting device is 11 bit times x 16 RTt cycles = 176 RTt cycles.

The maximum percent difference between the receiver count and the transmitter count of a fast 9-bit character with no errors is:

((176 – 170) /176) x 100 = 3.40%

### 20.4.6.6 Receiver Wakeup

To enable the SCI to ignore transmissions intended only for other receivers in multiple-receiver systems, the receiver can be put into a standby state. Setting the receiver wakeup bit, RWU, in SCI control register 2 (SCICR2) puts the receiver into standby state during which receiver interrupts are disabled.The SCI will still load the receive data into the SCIDRH/L registers, but it will not set the RDRF flag.

The transmitting device can address messages to selected receivers by including addressing information in the initial frame or frames of each message.

The WAKE bit in SCI control register 1 (SCICR1) determines how the SCI is brought out of the standby state to process an incoming message. The WAKE bit enables either idle line wakeup or address mark wakeup.

#### 20.4.6.6.1 Idle Input line Wakeup (WAKE = 0)

In this wakeup method, an idle condition on the RXD pin clears the RWU bit and wakes up the SCI. The initial frame or frames of every message contain addressing information. All receivers evaluate the addressing information, and receivers for which the message is addressed process the frames that follow. Any receiver for which a message is not addressed can set its RWU bit and return to the standby state. The RWU bit remains set and the receiver remains on standby until another idle character appears on the RXD pin.

Idle line wakeup requires that messages be separated by at least one idle character and that no message contains idle characters.

The idle character that wakes a receiver does not set the receiver idle bit, IDLE, or the receive data register full flag, RDRF.

The idle line type bit, ILT, determines whether the receiver begins counting logic 1s as idle character bits after the start bit or after the stop bit. ILT is in SCI control register 1 (SCICR1).

#### 20.4.6.6.2 Address Mark Wakeup (WAKE = 1)

In this wakeup method, a logic 1 in the most significant bit (MSB) position of a frame clears the RWU bit and wakes up the SCI. The logic 1 in the MSB position marks a frame as an address frame that contains addressing information. All receivers evaluate the addressing information, and the receivers for which the message is addressed process the frames that follow.Any receiver for which a message is not addressed can set its RWU bit and return to the standby state. The RWU bit remains set and the receiver remains on standby until another address frame appears on the RXD pin.

The logic 1 MSB of an address frame clears the receiver's RWU bit before the stop bit is received and sets the RDRF flag.

Address mark wakeup allows messages to contain idle characters but requires that the MSB be reserved for use in address frames.

**NOTE**

With the WAKE bit clear, setting the RWU bit after the RXD pin has been idle can cause the receiver to wake up immediately.

### 20.4.7 Single-Wire Operation

Normally, the SCI uses two pins for transmitting and receiving. In single-wire operation, the RXD pin is disconnected from the SCI. The SCI uses the TXD pin for both receiving and transmitting.

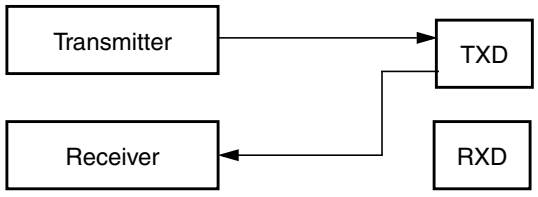



**Figure 20-30. Single-Wire Operation (LOOPS = 1, RSRC = 1)**

Enable single-wire operation by setting the LOOPS bit and the receiver source bit, RSRC, in SCI control register 1 (SCICR1). Setting the LOOPS bit disables the path from the RXD pin to the receiver. Setting the RSRC bit connects the TXD pin to the receiver. Both the transmitter and receiver must be enabled (TE = 1 and RE = 1).The TXDIR bit (SCISR2[1]) determines whether the TXD pin is going to be used as an input (TXDIR = 0) or an output (TXDIR = 1) in this mode of operation.

**NOTE**

In single-wire operation data from the TXD pin is inverted if RXPOL is set.

### 20.4.8 Loop Operation

In loop operation the transmitter output goes to the receiver input. The RXD pin is disconnected from the SCI.

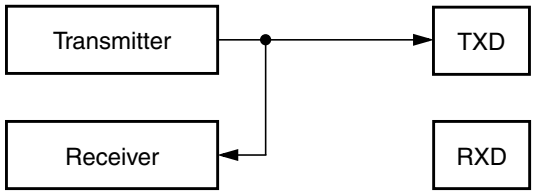



**Figure 20-31. Loop Operation (LOOPS = 1, RSRC = 0)**

Enable loop operation by setting the LOOPS bit and clearing the RSRC bit in SCI control register 1 (SCICR1). Setting the LOOPS bit disables the path from the RXD pin to the receiver. Clearing the RSRC bit connects the transmitter output to the receiver input. Both the transmitter and receiver must be enabled (TE = 1 and RE = 1).

**NOTE**

In loop operation data from the transmitter is not recognized by the receiver if RXPOL and TXPOL are not the same.

## 20.5 Initialization/Application Information

### 20.5.1 Reset Initialization

See Section 20.3.2, “Register Descriptions”.

### 20.5.2 Modes of Operation

#### 20.5.2.1 Run Mode

Normal mode of operation.

To initialize a SCI transmission, see Section 20.4.5.2, “Character Transmission”.

### 20.5.2.2 Wait Mode

SCI operation in wait mode depends on the state of the SCISWAI bit in the SCI control register 1 (SCICR1).

- If SCISWAI is clear, the SCI operates normally when the CPU is in wait mode.
- If SCISWAI is set, SCI clock generation ceases and the SCI module enters a power-conservation state when the CPU is in wait mode. Setting SCISWAI does not affect the state of the receiver enable bit, RE, or the transmitter enable bit, TE.

  If SCISWAI is set, any transmission or reception in progress stops at wait mode entry. The transmission or reception resumes when either an internal or external interrupt brings the CPU out of wait mode. Exiting wait mode by reset aborts any transmission or reception in progress and resets the SCI.

### 20.5.2.3 Stop Mode

The SCI is inactive during stop mode for reduced power consumption. The STOP instruction does not affect the SCI register states, but the SCI bus clock will be disabled. The SCI operation resumes from where it left off after an external interrupt brings the CPU out of stop mode. Exiting stop mode by reset aborts any transmission or reception in progress and resets the SCI.

The receive input active edge detect circuit is still active in stop mode. An active edge on the receive input can be used to bring the CPU out of stop mode.

## 20.5.3 Interrupt Operation

This section describes the interrupt originated by the SCI block.The MCU must service the interrupt requests. Table 20-20 lists the eight interrupt sources of the SCI.

**Table 20-20. SCI Interrupt Sources**

| Interrupt | Source | Local Enable | Description |
|---|---|---|---|
| TDRE | SCISR1[7] | TIE | Active high level. Indicates that a byte was transferred from SCIDRH/L to the transmit shift register. |
| TC | SCISR1[6] | TCIE | Active high level. Indicates that a transmit is complete. |
| RDRF | SCISR1[5] | RIE | Active high level. The RDRF interrupt indicates that received data is available in the SCI data register. |
| OR | SCISR1[3] | | Active high level. This interrupt indicates that an overrun condition has occurred. |
| IDLE | SCISR1[4] | ILIE | Active high level. Indicates that receiver input has become idle. |
| RXEDGIF | SCIASR1[7] | RXEDGIE | Active high level. Indicates that an active edge (falling for RXPOL = 0, rising for RXPOL = 1) was detected. |
| BERRIF | SCIASR1[1] | BERRIE | Active high level. Indicates that a mismatch between transmitted and received data in a single wire application has happened. |
| BKDIF | SCIASR1[0] | BRKDIE | Active high level. Indicates that a break character has been received. |

### 20.5.3.1 Description of Interrupt Operation

The SCI only originates interrupt requests. The following is a description of how the SCI makes a request and how the MCU should acknowledge that request. The interrupt vector offset and interrupt number are chip dependent. The SCI only has a single interrupt line (SCI Interrupt Signal, active high operation) and all the following interrupts, when generated, are ORed together and issued through that port.

#### 20.5.3.1.1 TDRE Description

The TDRE interrupt is set high by the SCI when the transmit shift register receives a byte from the SCI data register. A TDRE interrupt indicates that the transmit data register (SCIDRH/L) is empty and that a new byte can be written to the SCIDRH/L for transmission.Clear TDRE by reading SCI status register 1 with TDRE set and then writing to SCI data register low (SCIDRL).

#### 20.5.3.1.2 TC Description

The TC interrupt is set by the SCI when a transmission has been completed. Transmission is completed when all bits including the stop bit (if transmitted) have been shifted out and no data is queued to be transmitted. No stop bit is transmitted when sending a break character and the TC flag is set (providing there is no more data queued for transmission) when the break character has been shifted out. A TC interrupt indicates that there is no transmission in progress. TC is set high when the TDRE flag is set and no data, preamble, or break character is being transmitted. When TC is set, the TXD pin becomes idle (logic 1). Clear TC by reading SCI status register 1 (SCISR1) with TC set and then writing to SCI data register low (SCIDRL).TC is cleared automatically when data, preamble, or break is queued and ready to be sent.

#### 20.5.3.1.3 RDRF Description

The RDRF interrupt is set when the data in the receive shift register transfers to the SCI data register. A RDRF interrupt indicates that the received data has been transferred to the SCI data register and that the byte can now be read by the MCU. The RDRF interrupt is cleared by reading the SCI status register one (SCISR1) and then reading SCI data register low (SCIDRL).

#### 20.5.3.1.4 OR Description

The OR interrupt is set when software fails to read the SCI data register before the receive shift register receives the next frame. The newly acquired data in the shift register will be lost in this case, but the data already in the SCI data registers is not affected. The OR interrupt is cleared by reading the SCI status register one (SCISR1) and then reading SCI data register low (SCIDRL).

#### 20.5.3.1.5 IDLE Description

The IDLE interrupt is set when 10 consecutive logic 1s (if M = 0) or 11 consecutive logic 1s (if M = 1) appear on the receiver input. Once the IDLE is cleared, a valid frame must again set the RDRF flag before an idle condition can set the IDLE flag. Clear IDLE by reading SCI status register 1 (SCISR1) with IDLE set and then reading SCI data register low (SCIDRL).

#### 20.5.3.1.6 RXEDGIF Description

The RXEDGIF interrupt is set when an active edge (falling if RXPOL = 0, rising if RXPOL = 1) on the RXD pin is detected. Clear RXEDGIF by writing a “1” to the SCIASR1 SCI alternative status register 1.

#### 20.5.3.1.7 BERRIF Description

The BERRIF interrupt is set when a mismatch between the transmitted and the received data in a single wire application like LIN was detected. Clear BERRIF by writing a “1” to the SCIASR1 SCI alternative status register 1. This flag is also cleared if the bit error detect feature is disabled.

#### 20.5.3.1.8 BKDIF Description

The BKDIF interrupt is set when a break signal was received. Clear BKDIF by writing a “1” to the SCIASR1 SCI alternative status register 1. This flag is also cleared if break detect feature is disabled.

### 20.5.4 Recovery from Wait Mode

The SCI interrupt request can be used to bring the CPU out of wait mode.

### 20.5.5 Recovery from Stop Mode

An active edge on the receive input can be used to bring the CPU out of stop mode.

# Chapter 21
# Serial Peripheral Interface (S12SPIV5)

**Table 21-1. Revision History**

| Revision Number | Revision Date | Sections Affected | Description of Changes |
|---|---|---|---|
| V05.00 | 24 Mar 2005 | 21.3.2/21-765 | - Added 16-bit transfer width feature. |

## 21.1 Introduction

The SPI module allows a duplex, synchronous, serial communication between the MCU and peripheral devices. Software can poll the SPI status flags or the SPI operation can be interrupt driven.

### 21.1.1 Glossary of Terms

| | |
|---|---|
| SPI | Serial Peripheral Interface |
| $\overline{SS}$ | Slave Select |
| SCK | Serial Clock |
| MOSI | Master Output, Slave Input |
| MISO | Master Input, Slave Output |
| MOMI | Master Output, Master Input |
| SISO | Slave Input, Slave Output |

### 21.1.2 Features

The SPI includes these distinctive features:

- Master mode and slave mode
- Selectable 8 or 16-bit transfer width
- Bidirectional mode
- Slave select output
- Mode fault error flag with CPU interrupt capability
- Double-buffered data register
- Serial clock with programmable polarity and phase
- Control of SPI operation during wait mode

### 21.1.3 Modes of Operation

The SPI functions in three modes: run, wait, and stop.

- Run mode

  This is the basic mode of operation.
- Wait mode

  SPI operation in wait mode is a configurable low power mode, controlled by the SPISWAI bit located in the SPICR2 register. In wait mode, if the SPISWAI bit is clear, the SPI operates like in run mode. If the SPISWAI bit is set, the SPI goes into a power conservative state, with the SPI clock generation turned off. If the SPI is configured as a master, any transmission in progress stops, but is resumed after CPU goes into run mode. If the SPI is configured as a slave, reception and transmission of data continues, so that the slave stays synchronized to the master.
- Stop mode

  The SPI is inactive in stop mode for reduced power consumption. If the SPI is configured as a master, any transmission in progress stops, but is resumed after CPU goes into run mode. If the SPI is configured as a slave, reception and transmission of data continues, so that the slave stays synchronized to the master.

For a detailed description of operating modes, please refer to Section 21.4.7, “Low Power Mode Options”.

### 21.1.4 Block Diagram

Figure 21-1 gives an overview on the SPI architecture. The main parts of the SPI are status, control and data registers, shifter logic, baud rate generator, master/slave control logic, and port control logic.

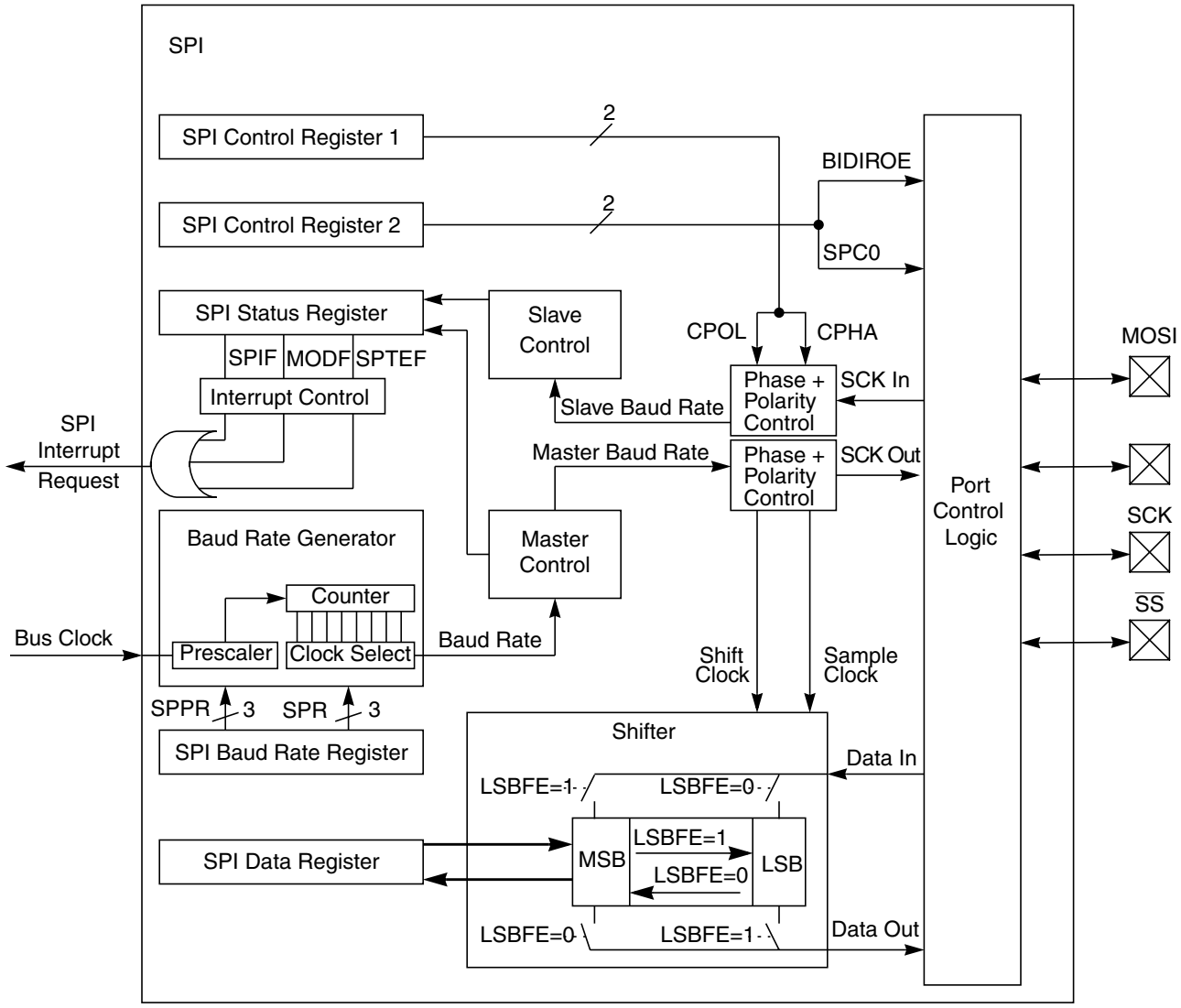



**Figure 21-1. SPI Block Diagram**

## 21.2 External Signal Description

This section lists the name and description of all ports including inputs and outputs that do, or may, connect off chip. The SPI module has a total of four external pins.

### 21.2.1 MOSI — Master Out/Slave In Pin

This pin is used to transmit data out of the SPI module when it is configured as a master and receive data when it is configured as slave.

### 21.2.2 MISO — Master In/Slave Out Pin

This pin is used to transmit data out of the SPI module when it is configured as a slave and receive data when it is configured as master.

### 21.2.3 $\overline{SS}$ — Slave Select Pin

This pin is used to output the select signal from the SPI module to another peripheral with which a data transfer is to take place when it is configured as a master and it is used as an input to receive the slave select signal when the SPI is configured as slave.

### 21.2.4 SCK — Serial Clock Pin

In master mode, this is the synchronous output clock. In slave mode, this is the synchronous input clock.

## 21.3 Memory Map and Register Definition

This section provides a detailed description of address space and registers used by the SPI.

### 21.3.1 Module Memory Map

The memory map for the SPI is given in Figure 21-2. The address listed for each register is the sum of a base address and an address offset. The base address is defined at the SoC level and the address offset is defined at the module level. Reads from the reserved bits return zeros and writes to the reserved bits have no effect.

| Register Name | | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|---|
| 0x0000 SPICR1 | R | SPIE | SPE | SPTIE | MSTR | CPOL | CPHA | SSOE | LSBFE |
| | W | | | | | | | | |
| 0x0001 SPICR2 | R | 0 | XFRW | 0 | MODFEN | BIDIROE | 0 | SPISWAI | SPC0 |
| | W | | | | | | | | |
| 0x0002 SPIBR | R | 0 | SPPR2 | SPPR1 | SPPR0 | 0 | SPR2 | SPR1 | SPR0 |
| | W | | | | | | | | |
| 0x0003 SPISR | R | SPIF | 0 | SPTEF | MODF | 0 | 0 | 0 | 0 |
| | W | | | | | | | | |
| 0x0004 SPIDRH | R | R15 | R14 | R13 | R12 | R11 | R10 | R9 | R8 |
| | W | T15 | T14 | T13 | T12 | T11 | T10 | T9 | T8 |
| 0x0005 SPIDRL | R | R7 | R6 | R5 | R4 | R3 | R2 | R1 | R0 |
| | W | T7 | T6 | T5 | T4 | T3 | T2 | T1 | T0 |
| 0x0006 Reserved | R | | | | | | | | |
| | W | | | | | | | | |
| 0x0007 Reserved | R | | | | | | | | |
| | W | | | | | | | | |

= Unimplemented or Reserved

**Figure 21-2. SPI Register Summary**

## 21.3.2 Register Descriptions

This section consists of register descriptions in address order. Each description includes a standard register diagram with an associated figure number. Details of register bit and field function follow the register diagrams, in bit order.

### 21.3.2.1 SPI Control Register 1 (SPICR1)

Module Base +0x0000

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | SPIE | SPE | SPTIE | MSTR | CPOL | CPHA | SSOE | LSBFE |
| Reset | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |

**Figure 21-3. SPI Control Register 1 (SPICR1)**

Read: Anytime

Write: Anytime

**Table 21-2. SPICR1 Field Descriptions**

| Field | Description |
|---|---|
| 7<br>SPIE | **SPI Interrupt Enable Bit** — This bit enables SPI interrupt requests, if SPIF or MODF status flag is set.<br>0 SPI interrupts disabled.<br>1 SPI interrupts enabled. |
| 6<br>SPE | **SPI System Enable Bit** — This bit enables the SPI system and dedicates the SPI port pins to SPI system functions. If SPE is cleared, SPI is disabled and forced into idle state, status bits in SPISR register are reset.<br>0 SPI disabled (lower power consumption).<br>1 SPI enabled, port pins are dedicated to SPI functions. |
| 5<br>SPTIE | **SPI Transmit Interrupt Enable** — This bit enables SPI interrupt requests, if SPTEF flag is set.<br>0 SPTEF interrupt disabled.<br>1 SPTEF interrupt enabled. |
| 4<br>MSTR | **SPI Master/Slave Mode Select Bit** — This bit selects whether the SPI operates in master or slave mode. Switching the SPI from master to slave or vice versa forces the SPI system into idle state.<br>0 SPI is in slave mode.<br>1 SPI is in master mode. |
| 3<br>CPOL | **SPI Clock Polarity Bit** — This bit selects an inverted or non-inverted SPI clock. To transmit data between SPI modules, the SPI modules must have identical CPOL values. In master mode, a change of this bit will abort a transmission in progress and force the SPI system into idle state.<br>0 Active-high clocks selected. In idle state SCK is low.<br>1 Active-low clocks selected. In idle state SCK is high. |
| 2<br>CPHA | **SPI Clock Phase Bit** — This bit is used to select the SPI clock format. In master mode, a change of this bit will abort a transmission in progress and force the SPI system into idle state.<br>0 Sampling of data occurs at odd edges (1,3,5,...) of the SCK clock.<br>1 Sampling of data occurs at even edges (2,4,6,...) of the SCK clock. |

**Table 21-2. SPICR1 Field Descriptions (continued)**

| Field | Description |
|---|---|
| 1<br>SSOE | **Slave Select Output Enable** — The $\overline{SS}$ output feature is enabled only in master mode, if MODFEN is set, by asserting the SSOE as shown in Table 21-3. In master mode, a change of this bit will abort a transmission in progress and force the SPI system into idle state. |
| 0<br>LSBFE | **LSB-First Enable** — This bit does not affect the position of the MSB and LSB in the data register. Reads and writes of the data register always have the MSB in the highest bit position. In master mode, a change of this bit will abort a transmission in progress and force the SPI system into idle state.<br>0 Data is transferred most significant bit first.<br>1 Data is transferred least significant bit first. |

**Table 21-3. $\overline{SS}$ Input / Output Selection**

| MODFEN | SSOE | Master Mode | Slave Mode |
|---|---|---|---|
| 0 | 0 | $\overline{SS}$ not used by SPI | $\overline{SS}$ input |
| 0 | 1 | $\overline{SS}$ not used by SPI | $\overline{SS}$ input |
| 1 | 0 | $\overline{SS}$ input with MODF feature | $\overline{SS}$ input |
| 1 | 1 | $\overline{SS}$ is slave select output | $\overline{SS}$ input |

### 21.3.2.2 SPI Control Register 2 (SPICR2)

Module Base +0x0001

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | 0 | XFRW | 0 | MODFEN | BIDIROE | 0 | SPISWAI | SPC0 |
| W | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented or Reserved

**Figure 21-4. SPI Control Register 2 (SPICR2)**

Read: Anytime

Write: Anytime; writes to the reserved bits have no effect

**Table 21-4. SPICR2 Field Descriptions**

| Field | Description |
|---|---|
| 6<br>XFRW | **Transfer Width —** This bit is used for selecting the data transfer width. If 8-bit transfer width is selected, SPIDRL becomes the dedicated data register and SPIDRH is unused. If 16-bit transfer width is selected, SPIDRH and SPIDRL form a 16-bit data register. Please refer to Section 21.3.2.4, “SPI Status Register (SPISR) for information about transmit/receive data handling and the interrupt flag clearing mechanism. In master mode, a change of this bit will abort a transmission in progress and force the SPI system into idle state.<br>0 8-bit Transfer Width (n = 8)(1)<br>1 16-bit Transfer Width (n = 16)[1] |
| 4<br>MODFEN | **Mode Fault Enable Bit** — This bit allows the MODF failure to be detected. If the SPI is in master mode and MODFEN is cleared, then the $\overline{SS}$ port pin is not used by the SPI. In slave mode, the $\overline{SS}$ is available only as an input regardless of the value of MODFEN. For an overview on the impact of the MODFEN bit on the $\overline{SS}$ port pin configuration, refer to Table 21-3. In master mode, a change of this bit will abort a transmission in progress and force the SPI system into idle state.<br>0 $\overline{SS}$ port pin is not used by the SPI.<br>1 $\overline{SS}$ port pin with MODF feature. |
| 3<br>BIDIROE | **Output Enable in the Bidirectional Mode of Operation** — This bit controls the MOSI and MISO output buffer of the SPI, when in bidirectional mode of operation (SPC0 is set). In master mode, this bit controls the output buffer of the MOSI port, in slave mode it controls the output buffer of the MISO port. In master mode, with SPC0 set, a change of this bit will abort a transmission in progress and force the SPI into idle state.<br>0 Output buffer disabled.<br>1 Output buffer enabled. |
| 1<br>SPISWAI | **SPI Stop in Wait Mode Bit** — This bit is used for power conservation while in wait mode.<br>0 SPI clock operates normally in wait mode.<br>1 Stop SPI clock generation when in wait mode. |
| 0<br>SPC0 | **Serial Pin Control Bit 0** — This bit enables bidirectional pin configurations as shown in Table 21-5. In master mode, a change of this bit will abort a transmission in progress and force the SPI system into idle state. |

1. n is used later in this document as a placeholder for the selected transfer width.

**Table 21-5. Bidirectional Pin Configurations**

| Pin Mode | SPC0 | BIDIROE | MISO | MOSI |
|---|---|---|---|---|
| **Master Mode of Operation** | | | | |
| Normal | 0 | X | Master In | Master Out |
| Bidirectional | 1 | 0 | MISO not used by SPI | Master In |
| | | 1 | | Master I/O |
| **Slave Mode of Operation** | | | | |
| Normal | 0 | X | Slave Out | Slave In |
| Bidirectional | 1 | 0 | Slave In | MOSI not used by SPI |
| | | 1 | Slave I/O | |

### 21.3.2.3 SPI Baud Rate Register (SPIBR)

Module Base +0x0002

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | 0 | SPPR2 | SPPR1 | SPPR0 | 0 | SPR2 | SPR1 | SPR0 |
| W | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented or Reserved

**Figure 21-5. SPI Baud Rate Register (SPIBR)**

Read: Anytime

Write: Anytime; writes to the reserved bits have no effect

**Table 21-6. SPIBR Field Descriptions**

| Field | Description |
|---|---|
| 6–4<br>SPPR[2:0] | **SPI Baud Rate Preselection Bits** — These bits specify the SPI baud rates as shown in Table 21-7. In master mode, a change of these bits will abort a transmission in progress and force the SPI system into idle state. |
| 2–0<br>SPR[2:0] | **SPI Baud Rate Selection Bits** — These bits specify the SPI baud rates as shown in Table 21-7. In master mode, a change of these bits will abort a transmission in progress and force the SPI system into idle state. |

The baud rate divisor equation is as follows:

$$\text{BaudRateDivisor} = (\text{SPPR} + 1) \bullet 2^{(\text{SPR} + 1)} \qquad \textit{Eqn. 21-1}$$

The baud rate can be calculated with the following equation:

$$\text{Baud Rate} = \text{BusClock} / \text{BaudRateDivisor} \qquad \textit{Eqn. 21-2}$$

**NOTE**

For maximum allowed baud rates, please refer to the SPI Electrical Specification in the Electricals chapter of this data sheet.

**Table 21-7. Example SPI Baud Rate Selection (25 MHz Bus Clock) (Sheet 1 of 3)**

| SPPR2 | SPPR1 | SPPR0 | SPR2 | SPR1 | SPR0 | Baud Rate Divisor | Baud Rate |
|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 2 | 12.5 Mbit/s |
| 0 | 0 | 0 | 0 | 0 | 1 | 4 | 6.25 Mbit/s |
| 0 | 0 | 0 | 0 | 1 | 0 | 8 | 3.125 Mbit/s |
| 0 | 0 | 0 | 0 | 1 | 1 | 16 | 1.5625 Mbit/s |
| 0 | 0 | 0 | 1 | 0 | 0 | 32 | 781.25 kbit/s |
| 0 | 0 | 0 | 1 | 0 | 1 | 64 | 390.63 kbit/s |
| 0 | 0 | 0 | 1 | 1 | 0 | 128 | 195.31 kbit/s |
| 0 | 0 | 0 | 1 | 1 | 1 | 256 | 97.66 kbit/s |
| 0 | 0 | 1 | 0 | 0 | 0 | 4 | 6.25 Mbit/s |
| 0 | 0 | 1 | 0 | 0 | 1 | 8 | 3.125 Mbit/s |
| 0 | 0 | 1 | 0 | 1 | 0 | 16 | 1.5625 Mbit/s |
| 0 | 0 | 1 | 0 | 1 | 1 | 32 | 781.25 kbit/s |

**Table 21-7. Example SPI Baud Rate Selection (25 MHz Bus Clock) (Sheet 2 of 3)**

| SPPR2 | SPPR1 | SPPR0 | SPR2 | SPR1 | SPR0 | Baud Rate Divisor | Baud Rate |
|---|---|---|---|---|---|---|---|
| 0 | 0 | 1 | 1 | 0 | 0 | 64 | 390.63 kbit/s |
| 0 | 0 | 1 | 1 | 0 | 1 | 128 | 195.31 kbit/s |
| 0 | 0 | 1 | 1 | 1 | 0 | 256 | 97.66 kbit/s |
| 0 | 0 | 1 | 1 | 1 | 1 | 512 | 48.83 kbit/s |
| 0 | 1 | 0 | 0 | 0 | 0 | 6 | 4.16667 Mbit/s |
| 0 | 1 | 0 | 0 | 0 | 1 | 12 | 2.08333 Mbit/s |
| 0 | 1 | 0 | 0 | 1 | 0 | 24 | 1.04167 Mbit/s |
| 0 | 1 | 0 | 0 | 1 | 1 | 48 | 520.83 kbit/s |
| 0 | 1 | 0 | 1 | 0 | 0 | 96 | 260.42 kbit/s |
| 0 | 1 | 0 | 1 | 0 | 1 | 192 | 130.21 kbit/s |
| 0 | 1 | 0 | 1 | 1 | 0 | 384 | 65.10 kbit/s |
| 0 | 1 | 0 | 1 | 1 | 1 | 768 | 32.55 kbit/s |
| 0 | 1 | 1 | 0 | 0 | 0 | 8 | 3.125 Mbit/s |
| 0 | 1 | 1 | 0 | 0 | 1 | 16 | 1.5625 Mbit/s |
| 0 | 1 | 1 | 0 | 1 | 0 | 32 | 781.25 kbit/s |
| 0 | 1 | 1 | 0 | 1 | 1 | 64 | 390.63 kbit/s |
| 0 | 1 | 1 | 1 | 0 | 0 | 128 | 195.31 kbit/s |
| 0 | 1 | 1 | 1 | 0 | 1 | 256 | 97.66 kbit/s |
| 0 | 1 | 1 | 1 | 1 | 0 | 512 | 48.83 kbit/s |
| 0 | 1 | 1 | 1 | 1 | 1 | 1024 | 24.41 kbit/s |
| 1 | 0 | 0 | 0 | 0 | 0 | 10 | 2.5 Mbit/s |
| 1 | 0 | 0 | 0 | 0 | 1 | 20 | 1.25 Mbit/s |
| 1 | 0 | 0 | 0 | 1 | 0 | 40 | 625 kbit/s |
| 1 | 0 | 0 | 0 | 1 | 1 | 80 | 312.5 kbit/s |
| 1 | 0 | 0 | 1 | 0 | 0 | 160 | 156.25 kbit/s |
| 1 | 0 | 0 | 1 | 0 | 1 | 320 | 78.13 kbit/s |
| 1 | 0 | 0 | 1 | 1 | 0 | 640 | 39.06 kbit/s |
| 1 | 0 | 0 | 1 | 1 | 1 | 1280 | 19.53 kbit/s |
| 1 | 0 | 1 | 0 | 0 | 0 | 12 | 2.08333 Mbit/s |
| 1 | 0 | 1 | 0 | 0 | 1 | 24 | 1.04167 Mbit/s |
| 1 | 0 | 1 | 0 | 1 | 0 | 48 | 520.83 kbit/s |
| 1 | 0 | 1 | 0 | 1 | 1 | 96 | 260.42 kbit/s |
| 1 | 0 | 1 | 1 | 0 | 0 | 192 | 130.21 kbit/s |
| 1 | 0 | 1 | 1 | 0 | 1 | 384 | 65.10 kbit/s |
| 1 | 0 | 1 | 1 | 1 | 0 | 768 | 32.55 kbit/s |
| 1 | 0 | 1 | 1 | 1 | 1 | 1536 | 16.28 kbit/s |
| 1 | 1 | 0 | 0 | 0 | 0 | 14 | 1.78571 Mbit/s |
| 1 | 1 | 0 | 0 | 0 | 1 | 28 | 892.86 kbit/s |
| 1 | 1 | 0 | 0 | 1 | 0 | 56 | 446.43 kbit/s |
| 1 | 1 | 0 | 0 | 1 | 1 | 112 | 223.21 kbit/s |
| 1 | 1 | 0 | 1 | 0 | 0 | 224 | 111.61 kbit/s |
| 1 | 1 | 0 | 1 | 0 | 1 | 448 | 55.80 kbit/s |

**Table 21-7. Example SPI Baud Rate Selection (25 MHz Bus Clock) (Sheet 3 of 3)**

| SPPR2 | SPPR1 | SPPR0 | SPR2 | SPR1 | SPR0 | Baud Rate Divisor | Baud Rate |
|---|---|---|---|---|---|---|---|
| 1 | 1 | 0 | 1 | 1 | 0 | 896 | 27.90 kbit/s |
| 1 | 1 | 0 | 1 | 1 | 1 | 1792 | 13.95 kbit/s |
| 1 | 1 | 1 | 0 | 0 | 0 | 16 | 1.5625 Mbit/s |
| 1 | 1 | 1 | 0 | 0 | 1 | 32 | 781.25 kbit/s |
| 1 | 1 | 1 | 0 | 1 | 0 | 64 | 390.63 kbit/s |
| 1 | 1 | 1 | 0 | 1 | 1 | 128 | 195.31 kbit/s |
| 1 | 1 | 1 | 1 | 0 | 0 | 256 | 97.66 kbit/s |
| 1 | 1 | 1 | 1 | 0 | 1 | 512 | 48.83 kbit/s |
| 1 | 1 | 1 | 1 | 1 | 0 | 1024 | 24.41 kbit/s |
| 1 | 1 | 1 | 1 | 1 | 1 | 2048 | 12.21 kbit/s |

### 21.3.2.4 SPI Status Register (SPISR)

Module Base +0x0003

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | SPIF | 0 | SPTEF | MODF | 0 | 0 | 0 | 0 |
| W | | | | | | | | |
| Reset | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented or Reserved

**Figure 21-6. SPI Status Register (SPISR)**

Read: Anytime

Write: Has no effect

**Table 21-8. SPISR Field Descriptions**

| Field | Description |
|---|---|
| 7<br>SPIF | **SPIF Interrupt Flag** — This bit is set after received data has been transferred into the SPI data register. For information about clearing SPIF Flag, please refer to Table 21-9.<br>0 Transfer not yet complete.<br>1 New data copied to SPIDR. |
| 5<br>SPTEF | **SPI Transmit Empty Interrupt Flag** — If set, this bit indicates that the transmit data register is empty. For information about clearing this bit and placing data into the transmit data register, please refer to Table 21-10.<br>0 SPI data register not empty.<br>1 SPI data register empty. |
| 4<br>MODF | **Mode Fault Flag** — This bit is set if the $\overline{SS}$ input becomes low while the SPI is configured as a master and mode fault detection is enabled, MODFEN bit of SPICR2 register is set. Refer to MODFEN bit description in Section 21.3.2.2, “SPI Control Register 2 (SPICR2)”. The flag is cleared automatically by a read of the SPI status register (with MODF set) followed by a write to the SPI control register 1.<br>0 Mode fault has not occurred.<br>1 Mode fault has occurred. |

**Table 21-9. SPIF Interrupt Flag Clearing Sequence**

| XFRW Bit | SPIF Interrupt Flag Clearing Sequence | | | |
|---|---|---|---|---|
| 0 | Read SPISR with SPIF == 1 | **then** | Read SPIDRL | |
| 1 | Read SPISR with SPIF == 1 | **then** | Byte Read SPIDRL (1) | |
| | | | **or** | |
| | | | Byte Read SPIDRH (2) | Byte Read SPIDRL |
| | | | **or** | |
| | | | Word Read (SPIDRH:SPIDRL) | |

1. Data in SPIDRH is lost in this case.
2. SPIDRH can be read repeatedly without any effect on SPIF. SPIF Flag is cleared only by the read of SPIDRL after reading SPISR with SPIF == 1.

**Table 21-10. SPTEF Interrupt Flag Clearing Sequence**

| XFRW Bit | SPTEF Interrupt Flag Clearing Sequence | | | |
|---|---|---|---|---|
| 0 | Read SPISR with SPTEF == 1 | **then** | Write to SPIDRL (1) | |
| 1 | Read SPISR with SPTEF == 1 | **then** | Byte Write to SPIDRL 1(2) | |
| | | | **or** | |
| | | | Byte Write to SPIDRH 1(3) | Byte Write to SPIDRL 1 |
| | | | **or** | |
| | | | Word Write to (SPIDRH:SPIDRL) 1 | |

1. Any write to SPIDRH or SPIDRL with SPTEF == 0 is effectively ignored.
2. Data in SPIDRH is undefined in this case.
3. SPIDRH can be written repeatedly without any effect on SPTEF. SPTEF Flag is cleared only by writing to SPIDRL after reading SPISR with SPTEF == 1.

### 21.3.2.5 SPI Data Register (SPIDR = SPIDRH:SPIDRL)

Module Base +0x0004

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | R15 | R14 | R13 | R12 | R11 | R10 | R9 | R8 |
| W | T15 | T14 | T13 | T12 | T11 | T10 | T9 | T8 |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 21-7. SPI Data Register High (SPIDRH)**

Module Base +0x0005

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | R7 | R6 | R5 | R4 | R3 | R2 | R1 | R0 |
| W | T7 | T6 | T5 | T4 | T3 | T2 | T1 | T0 |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 21-8. SPI Data Register Low (SPIDRL)**

Read: Anytime; read data only valid when SPIF is set

Write: Anytime

The SPI data register is both the input and output register for SPI data. A write to this register allows data to be queued and transmitted. For an SPI configured as a master, queued data is transmitted immediately after the previous transmission has completed. The SPI transmitter empty flag SPTEF in the SPISR register indicates when the SPI data register is ready to accept new data.

Received data in the SPIDR is valid when SPIF is set.

If SPIF is cleared and data has been received, the received data is transferred from the receive shift register to the SPIDR and SPIF is set.

If SPIF is set and not serviced, and a second data value has been received, the second received data is kept as valid data in the receive shift register until the start of another transmission. The data in the SPIDR does not change.

If SPIF is set and valid data is in the receive shift register, and SPIF is serviced before the start of a third transmission, the data in the receive shift register is transferred into the SPIDR and SPIF remains set (see Figure 21-9).

If SPIF is set and valid data is in the receive shift register, and SPIF is serviced after the start of a third transmission, the data in the receive shift register has become invalid and is not transferred into the SPIDR (see Figure 21-10).

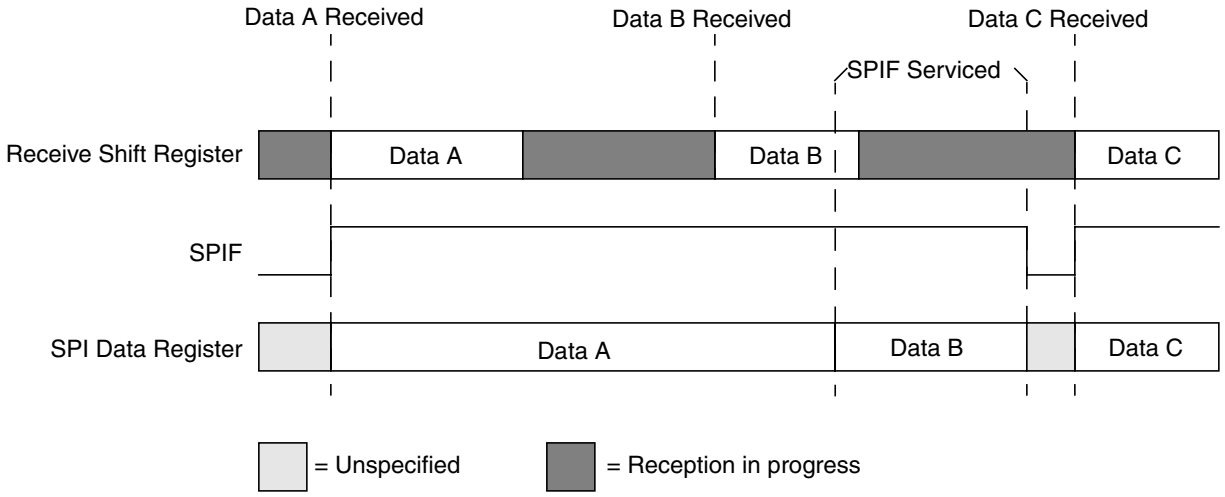



**Figure 21-9. Reception with SPIF serviced in Time**

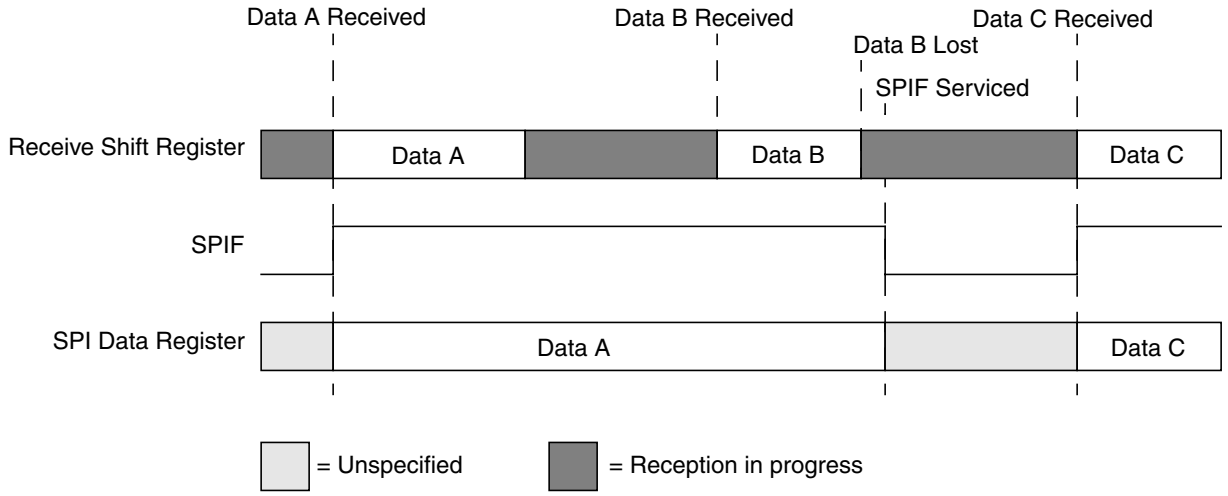



**Figure 21-10. Reception with SPIF serviced too late**

## 21.4 Functional Description

The SPI module allows a duplex, synchronous, serial communication between the MCU and peripheral devices. Software can poll the SPI status flags or SPI operation can be interrupt driven.

The SPI system is enabled by setting the SPI enable (SPE) bit in SPI control register 1. While SPE is set, the four associated SPI port pins are dedicated to the SPI function as:

- Slave select ($\overline{SS}$)
- Serial clock (SCK)
- Master out/slave in (MOSI)
- Master in/slave out (MISO)

The main element of the SPI system is the SPI data register. The n-bit[1] data register in the master and the n-bit[1] data register in the slave are linked by the MOSI and MISO pins to form a distributed 2n-bit[1] register. When a data transfer operation is performed, this 2n-bit[1] register is serially shifted n[1] bit positions by the S-clock from the master, so data is exchanged between the master and the slave. Data written to the master SPI data register becomes the output data for the slave, and data read from the master SPI data register after a transfer operation is the input data from the slave.

A read of SPISR with SPTEF = 1 followed by a write to SPIDR puts data into the transmit data register. When a transfer is complete and SPIF is cleared, received data is moved into the receive data register. This data register acts as the SPI receive data register for reads and as the SPI transmit data register for writes. A common SPI data register address is shared for reading data from the read data buffer and for writing data to the transmit data register.

The clock phase control bit (CPHA) and a clock polarity control bit (CPOL) in the SPI control register 1 (SPICR1) select one of four possible clock formats to be used by the SPI system. The CPOL bit simply selects a non-inverted or inverted clock. The CPHA bit is used to accommodate two fundamentally different protocols by sampling data on odd numbered SCK edges or on even numbered SCK edges (see Section 21.4.3, “Transmission Formats”).

The SPI can be configured to operate as a master or as a slave. When the MSTR bit in SPI control register1 is set, master mode is selected, when the MSTR bit is clear, slave mode is selected.

**NOTE**

A change of CPOL or MSTR bit while there is a received byte pending in the receive shift register will destroy the received byte and must be avoided.

### 21.4.1 Master Mode

The SPI operates in master mode when the MSTR bit is set. Only a master SPI module can initiate transmissions. A transmission begins by writing to the master SPI data register. If the shift register is empty, data immediately transfers to the shift register. Data begins shifting out on the MOSI pin under the control of the serial clock.

- Serial clock

  The SPR2, SPR1, and SPR0 baud rate selection bits, in conjunction with the SPPR2, SPPR1, and SPPR0 baud rate preselection bits in the SPI baud rate register, control the baud rate generator and determine the speed of the transmission. The SCK pin is the SPI clock output. Through the SCK pin, the baud rate generator of the master controls the shift register of the slave peripheral.
- MOSI, MISO pin

  In master mode, the function of the serial data output pin (MOSI) and the serial data input pin (MISO) is determined by the SPC0 and BIDIROE control bits.
- $\overline{SS}$ pin

  If MODFEN and SSOE are set, the $\overline{SS}$ pin is configured as slave select output. The $\overline{SS}$ output becomes low during each transmission and is high when the SPI is in idle state.

  If MODFEN is set and SSOE is cleared, the $\overline{SS}$ pin is configured as input for detecting mode fault error. If the $\overline{SS}$ input becomes low this indicates a mode fault error where another master tries to

1. n depends on the selected transfer width, please refer to Section 21.3.2.2, “SPI Control Register 2 (SPICR2)

drive the MOSI and SCK lines. In this case, the SPI immediately switches to slave mode, by clearing the MSTR bit and also disables the slave output buffer MISO (or SISO in bidirectional mode). So the result is that all outputs are disabled and SCK, MOSI, and MISO are inputs. If a transmission is in progress when the mode fault occurs, the transmission is aborted and the SPI is forced into idle state.

This mode fault error also sets the mode fault (MODF) flag in the SPI status register (SPISR). If the SPI interrupt enable bit (SPIE) is set when the MODF flag becomes set, then an SPI interrupt sequence is also requested.

When a write to the SPI data register in the master occurs, there is a half SCK-cycle delay. After the delay, SCK is started within the master. The rest of the transfer operation differs slightly, depending on the clock format specified by the SPI clock phase bit, CPHA, in SPI control register 1 (see Section 21.4.3, “Transmission Formats”).

**NOTE**

A change of the bits CPOL, CPHA, SSOE, LSBFE, XFRW, MODFEN, SPC0, or BIDIROE with SPC0 set, SPPR2-SPPR0 and SPR2-SPR0 in master mode will abort a transmission in progress and force the SPI into idle state. The remote slave cannot detect this, therefore the master must ensure that the remote slave is returned to idle state.

## 21.4.2 Slave Mode

The SPI operates in slave mode when the MSTR bit in SPI control register 1 is clear.

- Serial clock

  In slave mode, SCK is the SPI clock input from the master.

- MISO, MOSI pin

  In slave mode, the function of the serial data output pin (MISO) and serial data input pin (MOSI) is determined by the SPC0 bit and BIDIROE bit in SPI control register 2.

- $\overline{SS}$ pin

  The $\overline{SS}$ pin is the slave select input. Before a data transmission occurs, the $\overline{SS}$ pin of the slave SPI must be low. $\overline{SS}$ must remain low until the transmission is complete. If $\overline{SS}$ goes high, the SPI is forced into idle state.

  The $\overline{SS}$ input also controls the serial data output pin, if $\overline{SS}$ is high (not selected), the serial data output pin is high impedance, and, if $\overline{SS}$ is low, the first bit in the SPI data register is driven out of the serial data output pin. Also, if the slave is not selected ($\overline{SS}$ is high), then the SCK input is ignored and no internal shifting of the SPI shift register occurs.

  Although the SPI is capable of duplex operation, some SPI peripherals are capable of only receiving SPI data in a slave mode. For these simpler devices, there is no serial data out pin.

**NOTE**

When peripherals with duplex capability are used, take care not to simultaneously enable two receivers whose serial outputs drive the same system slave’s serial data output line.

As long as no more than one slave device drives the system slave's serial data output line, it is possible for several slaves to receive the same transmission from a master, although the master would not receive return information from all of the receiving slaves.

If the CPHA bit in SPI control register 1 is clear, odd numbered edges on the SCK input cause the data at the serial data input pin to be latched. Even numbered edges cause the value previously latched from the serial data input pin to shift into the LSB or MSB of the SPI shift register, depending on the LSBFE bit.

If the CPHA bit is set, even numbered edges on the SCK input cause the data at the serial data input pin to be latched. Odd numbered edges cause the value previously latched from the serial data input pin to shift into the LSB or MSB of the SPI shift register, depending on the LSBFE bit.

When CPHA is set, the first edge is used to get the first data bit onto the serial data output pin. When CPHA is clear and the $\overline{SS}$ input is low (slave selected), the first bit of the SPI data is driven out of the serial data output pin. After the nth[1] shift, the transfer is considered complete and the received data is transferred into the SPI data register. To indicate transfer is complete, the SPIF flag in the SPI status register is set.

**NOTE**

A change of the bits CPOL, CPHA, SSOE, LSBFE, MODFEN, SPC0, or BIDIROE with SPC0 set in slave mode will corrupt a transmission in progress and must be avoided.

## 21.4.3 Transmission Formats

During an SPI transmission, data is transmitted (shifted out serially) and received (shifted in serially) simultaneously. The serial clock (SCK) synchronizes shifting and sampling of the information on the two serial data lines. A slave select line allows selection of an individual slave SPI device; slave devices that are not selected do not interfere with SPI bus activities. Optionally, on a master SPI device, the slave select line can be used to indicate multiple-master bus contention.

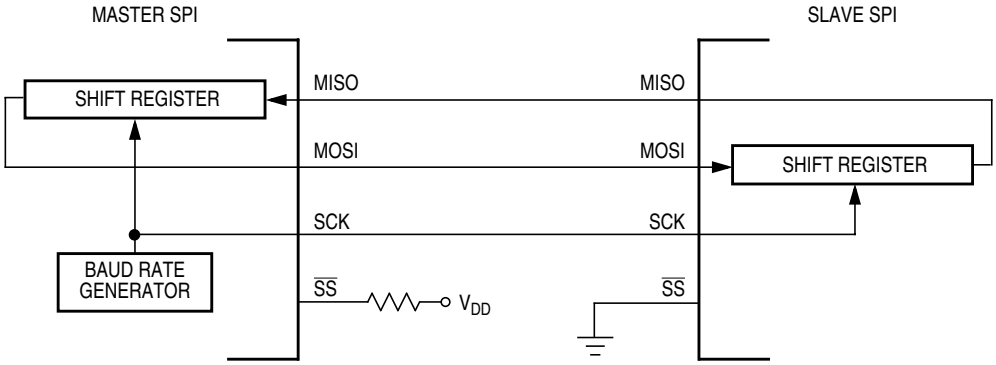



**Figure 21-11. Master/Slave Transfer Block Diagram**

### 21.4.3.1 Clock Phase and Polarity Controls

Using two bits in the SPI control register 1, software selects one of four combinations of serial clock phase and polarity.

1. n depends on the selected transfer width, please refer to Section 21.3.2.2, "SPI Control Register 2 (SPICR2)

The CPOL clock polarity control bit specifies an active high or low clock and has no significant effect on the transmission format.

The CPHA clock phase control bit selects one of two fundamentally different transmission formats.

Clock phase and polarity should be identical for the master SPI device and the communicating slave device. In some cases, the phase and polarity are changed between transmissions to allow a master device to communicate with peripheral slaves having different requirements.

### 21.4.3.2 CPHA = 0 Transfer Format

The first edge on the SCK line is used to clock the first data bit of the slave into the master and the first data bit of the master into the slave. In some peripherals, the first bit of the slave's data is available at the slave's data out pin as soon as the slave is selected. In this format, the first SCK edge is issued a half cycle after $\overline{SS}$ has become low.

A half SCK cycle later, the second edge appears on the SCK line. When this second edge occurs, the value previously latched from the serial data input pin is shifted into the LSB or MSB of the shift register, depending on LSBFE bit.

After this second edge, the next bit of the SPI master data is transmitted out of the serial data output pin of the master to the serial input pin on the slave. This process continues for a total of 16 edges on the SCK line, with data being latched on odd numbered edges and shifted on even numbered edges.

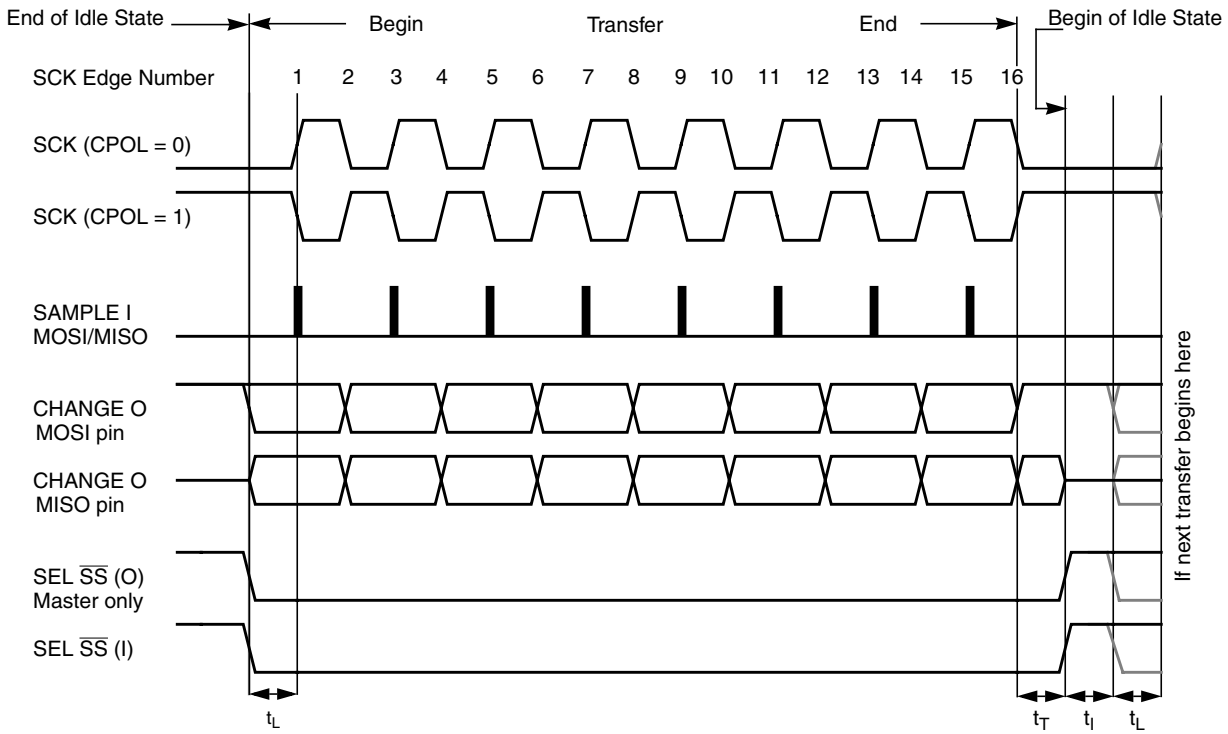
Data reception is double buffered. Data is shifted serially into the SPI shift register during the transfer and is transferred to the parallel SPI data register after the last bit is shifted in.

After $2n^1$ (last) SCK edges:

- Data that was previously in the master SPI data register should now be in the slave data register and the data that was in the slave data register should be in the master.
- The SPIF flag in the SPI status register is set, indicating that the transfer is complete.

Figure 21-12 is a timing diagram of an SPI transfer where CPHA = 0. SCK waveforms are shown for CPOL = 0 and CPOL = 1. The diagram may be interpreted as a master or slave timing diagram because the SCK, MISO, and MOSI pins are connected directly between the master and the slave. The MISO signal is the output from the slave and the MOSI signal is the output from the master. The $\overline{SS}$ pin of the master must be either high or reconfigured as a general-purpose output not affecting the SPI.

1. n depends on the selected transfer width, please refer to Section 21.3.2.2, "SPI Control Register 2 (SPICR2)




$t_L$ = Minimum leading time before the first SCK edge

$t_T$ = Minimum trailing time after the last SCK edge

$t_I$ = Minimum idling time between transfers (minimum $\overline{SS}$ high time)

$t_L$, $t_T$, and $t_I$ are guaranteed for the master mode and required for the slave mode.

**Figure 21-12. SPI Clock Format 0 (CPHA = 0), with 8-bit Transfer Width selected (XFRW = 0)**

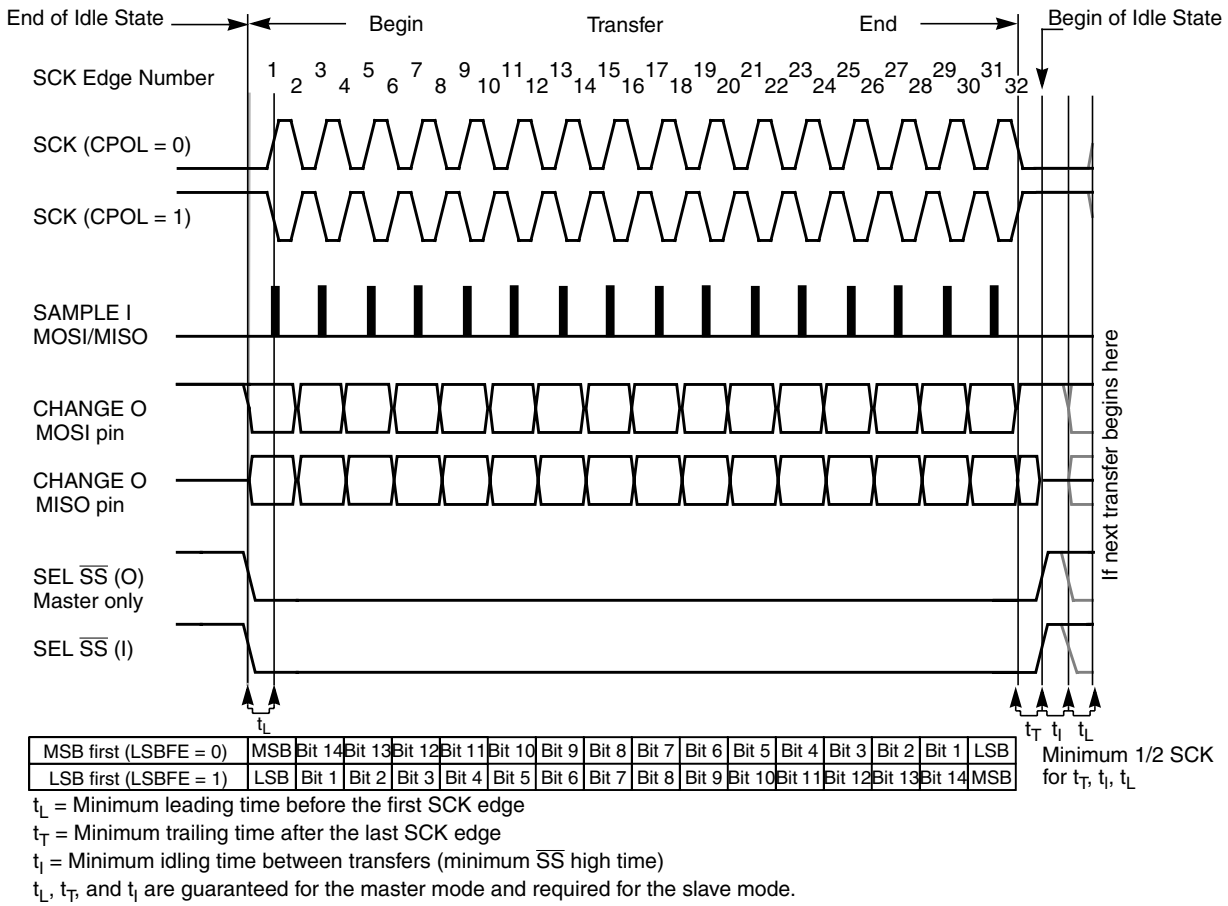



**Figure 21-13. SPI Clock Format 0 (CPHA = 0), with 16-Bit Transfer Width selected (XFRW = 1)**

In slave mode, if the $\overline{SS}$ line is not deasserted between the successive transmissions then the content of the SPI data register is not transmitted; instead the last received data is transmitted. If the $\overline{SS}$ line is deasserted for at least minimum idle time (half SCK cycle) between successive transmissions, then the content of the SPI data register is transmitted.

In master mode, with slave select output enabled the $\overline{SS}$ line is always deasserted and reasserted between successive transfers for at least minimum idle time.

### 21.4.3.3 CPHA = 1 Transfer Format

Some peripherals require the first SCK edge before the first data bit becomes available at the data out pin, the second edge clocks data into the system. In this format, the first SCK edge is issued by setting the CPHA bit at the beginning of the n[1]-cycle transfer operation.

The first edge of SCK occurs immediately after the half SCK clock cycle synchronization delay. This first edge commands the slave to transfer its first data bit to the serial data input pin of the master.

A half SCK cycle later, the second edge appears on the SCK pin. This is the latching edge for both the master and slave.

1. n depends on the selected transfer width, please refer to Section 21.3.2.2, “SPI Control Register 2 (SPICR2)

When the third edge occurs, the value previously latched from the serial data input pin is shifted into the LSB or MSB of the SPI shift register, depending on LSBFE bit. After this edge, the next bit of the master data is coupled out of the serial data output pin of the master to the serial input pin on the slave.

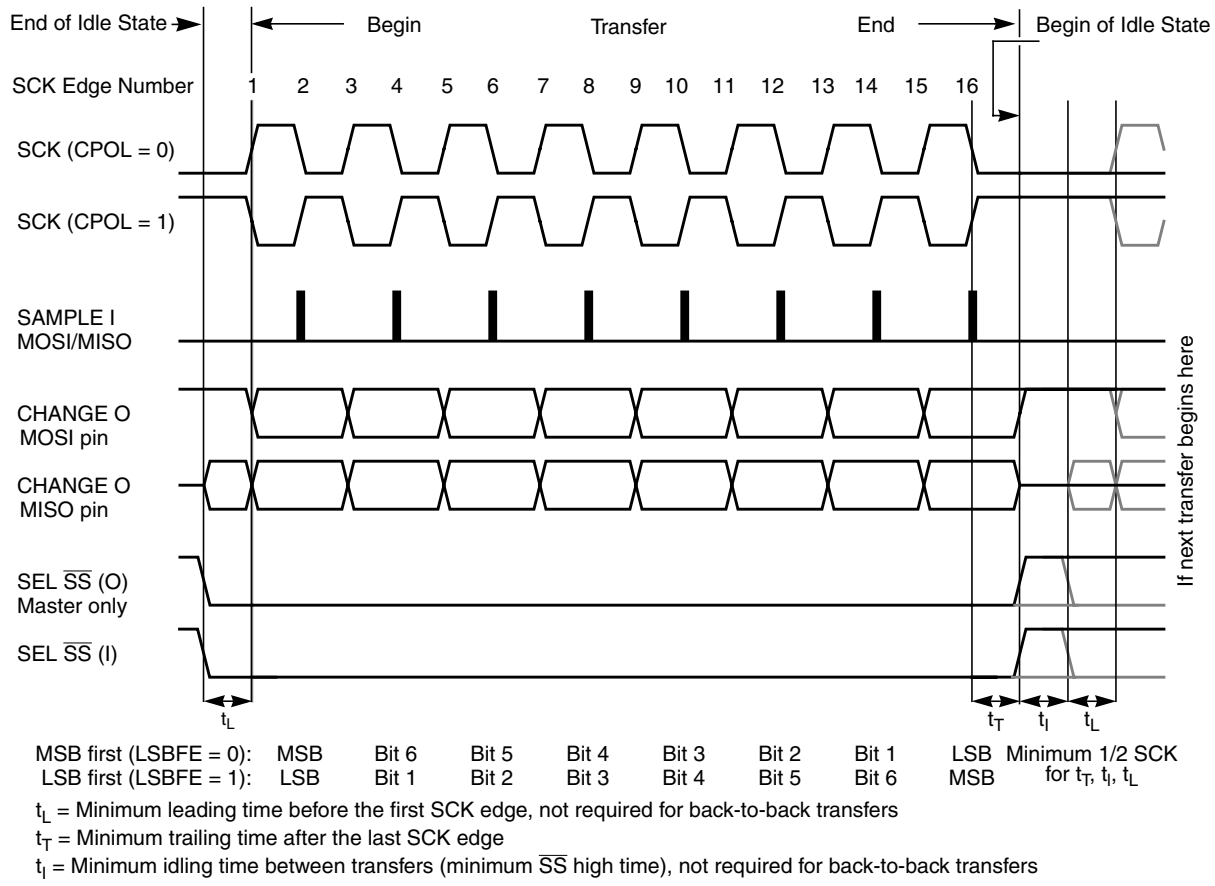
This process continues for a total of n[1] edges on the SCK line with data being latched on even numbered edges and shifting taking place on odd numbered edges.

Data reception is double buffered, data is serially shifted into the SPI shift register during the transfer and is transferred to the parallel SPI data register after the last bit is shifted in.

After 2n[1] SCK edges:

- Data that was previously in the SPI data register of the master is now in the data register of the slave, and data that was in the data register of the slave is in the master.
- The SPIF flag bit in SPISR is set indicating that the transfer is complete.

Figure 21-14 shows two clocking variations for CPHA = 1. The diagram may be interpreted as a master or slave timing diagram because the SCK, MISO, and MOSI pins are connected directly between the master and the slave. The MISO signal is the output from the slave, and the MOSI signal is the output from the master. The $\overline{SS}$ line is the slave select input to the slave. The $\overline{SS}$ pin of the master must be either high or reconfigured as a general-purpose output not affecting the SPI.




$t_L$ = Minimum leading time before the first SCK edge, not required for back-to-back transfers

$t_T$ = Minimum trailing time after the last SCK edge

$t_I$ = Minimum idling time between transfers (minimum $\overline{SS}$ high time), not required for back-to-back transfers

**Figure 21-14. SPI Clock Format 1 (CPHA = 1), with 8-Bit Transfer Width selected (XFRW = 0)**

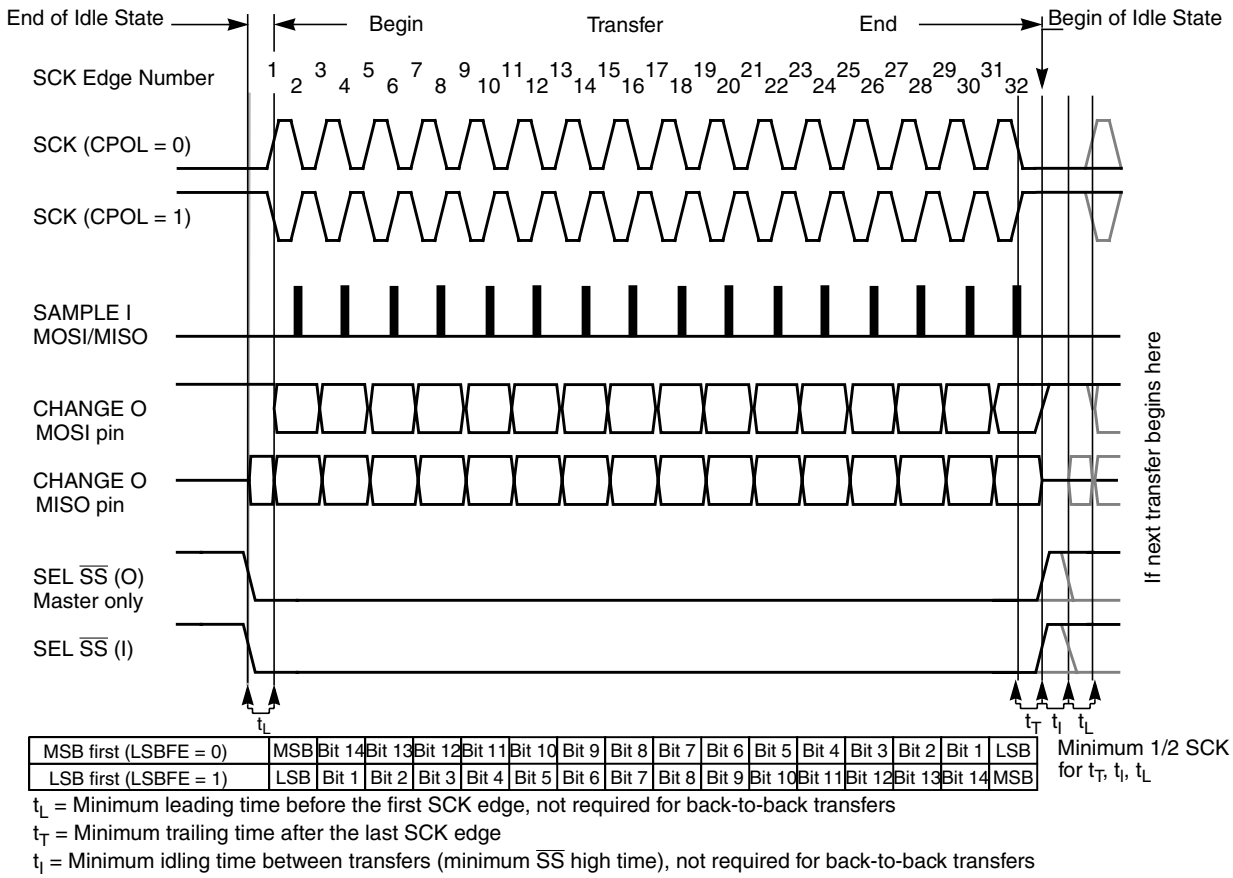



**Figure 21-15. SPI Clock Format 1 (CPHA = 1), with 16-Bit Transfer Width selected (XFRW = 1)**

The $\overline{SS}$ line can remain active low between successive transfers (can be tied low at all times). This format is sometimes preferred in systems having a single fixed master and a single slave that drive the MISO data line.

- Back-to-back transfers in master mode

  In master mode, if a transmission has completed and new data is available in the SPI data register, this data is sent out immediately without a trailing and minimum idle time.

The SPI interrupt request flag (SPIF) is common to both the master and slave modes. SPIF gets set one half SCK cycle after the last SCK edge.

## 21.4.4 SPI Baud Rate Generation

Baud rate generation consists of a series of divider stages. Six bits in the SPI baud rate register (SPPR2, SPPR1, SPPR0, SPR2, SPR1, and SPR0) determine the divisor to the SPI module clock which results in the SPI baud rate.

The SPI clock rate is determined by the product of the value in the baud rate preselection bits (SPPR2–SPPR0) and the value in the baud rate selection bits (SPR2–SPR0). The module clock divisor equation is shown in Equation 21-3.

$$\text{BaudRateDivisor} = (\text{SPPR} + 1) \bullet 2^{(\text{SPR} + 1)} \qquad \textit{Eqn. 21-3}$$

When all bits are clear (the default condition), the SPI module clock is divided by 2. When the selection bits (SPR2–SPR0) are 001 and the preselection bits (SPPR2–SPPR0) are 000, the module clock divisor becomes 4. When the selection bits are 010, the module clock divisor becomes 8, etc.

When the preselection bits are 001, the divisor determined by the selection bits is multiplied by 2. When the preselection bits are 010, the divisor is multiplied by 3, etc. See Table 21-7 for baud rate calculations for all bit conditions, based on a 25 MHz bus clock. The two sets of selects allows the clock to be divided by a non-power of two to achieve other baud rates such as divide by 6, divide by 10, etc.

The baud rate generator is activated only when the SPI is in master mode and a serial transfer is taking place. In the other cases, the divider is disabled to decrease $I_{DD}$ current.

**NOTE**

For maximum allowed baud rates, please refer to the SPI Electrical Specification in the Electricals chapter of this data sheet.

## 21.4.5 Special Features

### 21.4.5.1 $\overline{SS}$ Output

The $\overline{SS}$ output feature automatically drives the $\overline{SS}$ pin low during transmission to select external devices and drives it high during idle to deselect external devices. When $\overline{SS}$ output is selected, the $\overline{SS}$ output pin is connected to the $\overline{SS}$ input pin of the external device.

The $\overline{SS}$ output is available only in master mode during normal SPI operation by asserting SSOE and MODFEN bit as shown in Table 21-3.

The mode fault feature is disabled while $\overline{SS}$ output is enabled.

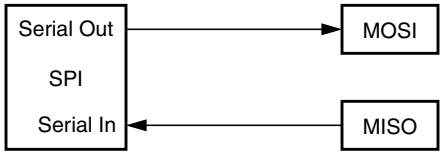
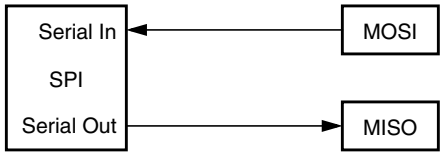
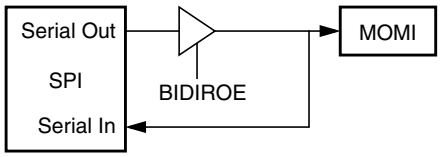
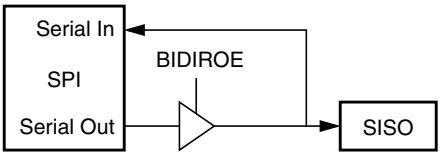
**NOTE**

Care must be taken when using the $\overline{SS}$ output feature in a multimaster system because the mode fault feature is not available for detecting system errors between masters.

### 21.4.5.2 Bidirectional Mode (MOMI or SISO)

The bidirectional mode is selected when the SPC0 bit is set in SPI control register 2 (see Table 21-11). In this mode, the SPI uses only one serial data pin for the interface with external device(s). The MSTR bit decides which pin to use. The MOSI pin becomes the serial data I/O (MOMI) pin for the master mode, and the MISO pin becomes serial data I/O (SISO) pin for the slave mode. The MISO pin in master mode and MOSI pin in slave mode are not used by the SPI.

**Table 21-11. Normal Mode and Bidirectional Mode**

| When SPE = 1 | Master Mode MSTR = 1 | Slave Mode MSTR = 0 |
|---|---|---|
| **Normal Mode SPC0 = 0** |  |  |
| **Bidirectional Mode SPC0 = 1** |  |  |

The direction of each serial I/O pin depends on the BIDIROE bit. If the pin is configured as an output, serial data from the shift register is driven out on the pin. The same pin is also the serial input to the shift register.

- The SCK is output for the master mode and input for the slave mode.
- The $\overline{SS}$ is the input or output for the master mode, and it is always the input for the slave mode.
- The bidirectional mode does not affect SCK and $\overline{SS}$ functions.

**NOTE**

In bidirectional master mode, with mode fault enabled, both data pins MISO and MOSI can be occupied by the SPI, though MOSI is normally used for transmissions in bidirectional mode and MISO is not used by the SPI. If a mode fault occurs, the SPI is automatically switched to slave mode. In this case MISO becomes occupied by the SPI and MOSI is not used. This must be considered, if the MISO pin is used for another purpose.

## 21.4.6 Error Conditions

The SPI has one error condition:

- Mode fault error

### 21.4.6.1 Mode Fault Error

If the $\overline{SS}$ input becomes low while the SPI is configured as a master, it indicates a system error where more than one master may be trying to drive the MOSI and SCK lines simultaneously. This condition is not permitted in normal operation, the MODF bit in the SPI status register is set automatically, provided the MODFEN bit is set.

In the special case where the SPI is in master mode and MODFEN bit is cleared, the $\overline{SS}$ pin is not used by the SPI. In this special case, the mode fault error function is inhibited and MODF remains cleared. In case

the SPI system is configured as a slave, the $\overline{SS}$ pin is a dedicated input pin. Mode fault error doesn't occur in slave mode.

If a mode fault error occurs, the SPI is switched to slave mode, with the exception that the slave output buffer is disabled. So SCK, MISO, and MOSI pins are forced to be high impedance inputs to avoid any possibility of conflict with another output driver. A transmission in progress is aborted and the SPI is forced into idle state.

If the mode fault error occurs in the bidirectional mode for a SPI system configured in master mode, output enable of the MOMI (MOSI in bidirectional mode) is cleared if it was set. No mode fault error occurs in the bidirectional mode for SPI system configured in slave mode.

The mode fault flag is cleared automatically by a read of the SPI status register (with MODF set) followed by a write to SPI control register 1. If the mode fault flag is cleared, the SPI becomes a normal master or slave again.

**NOTE**

If a mode fault error occurs and a received data byte is pending in the receive shift register, this data byte will be lost.

## 21.4.7 Low Power Mode Options

### 21.4.7.1 SPI in Run Mode

In run mode with the SPI system enable (SPE) bit in the SPI control register clear, the SPI system is in a low-power, disabled state. SPI registers remain accessible, but clocks to the core of this module are disabled.

### 21.4.7.2 SPI in Wait Mode

SPI operation in wait mode depends upon the state of the SPISWAI bit in SPI control register 2.

- If SPISWAI is clear, the SPI operates normally when the CPU is in wait mode
- If SPISWAI is set, SPI clock generation ceases and the SPI module enters a power conservation state when the CPU is in wait mode.
  - If SPISWAI is set and the SPI is configured for master, any transmission and reception in progress stops at wait mode entry. The transmission and reception resumes when the SPI exits wait mode.
  - If SPISWAI is set and the SPI is configured as a slave, any transmission and reception in progress continues if the SCK continues to be driven from the master. This keeps the slave synchronized to the master and the SCK.

    If the master transmits several bytes while the slave is in wait mode, the slave will continue to send out bytes consistent with the operation mode at the start of wait mode (i.e., if the slave is currently sending its SPIDR to the master, it will continue to send the same byte. Else if the slave is currently sending the last received byte from the master, it will continue to send each previous master byte).

**NOTE**

Care must be taken when expecting data from a master while the slave is in wait or stop mode. Even though the shift register will continue to operate, the rest of the SPI is shut down (i.e., a SPIF interrupt will **not** be generated until exiting stop or wait mode). Also, the byte from the shift register will not be copied into the SPIDR register until after the slave SPI has exited wait or stop mode. In slave mode, a received byte pending in the receive shift register will be lost when entering wait or stop mode. An SPIF flag and SPIDR copy is generated only if wait mode is entered or exited during a tranmission. If the slave enters wait mode in idle mode and exits wait mode in idle mode, neither a SPIF nor a SPIDR copy will occur.

#### 21.4.7.3 SPI in Stop Mode

Stop mode is dependent on the system. The SPI enters stop mode when the module clock is disabled (held high or low). If the SPI is in master mode and exchanging data when the CPU enters stop mode, the transmission is frozen until the CPU exits stop mode. After stop, data to and from the external SPI is exchanged correctly. In slave mode, the SPI will stay synchronized with the master.

The stop mode is not dependent on the SPISWAI bit.

#### 21.4.7.4 Reset

The reset values of registers and signals are described in Section 21.3, “Memory Map and Register Definition”, which details the registers and their bit fields.

- If a data transmission occurs in slave mode after reset without a write to SPIDR, it will transmit garbage, or the data last received from the master before the reset.
- Reading from the SPIDR after reset will always read zeros.

#### 21.4.7.5 Interrupts

The SPI only originates interrupt requests when SPI is enabled (SPE bit in SPICR1 set). The following is a description of how the SPI makes a request and how the MCU should acknowledge that request. The interrupt vector offset and interrupt priority are chip dependent.

The interrupt flags MODF, SPIF, and SPTEF are logically ORed to generate an interrupt request.

##### 21.4.7.5.1 MODF

MODF occurs when the master detects an error on the $\overline{SS}$ pin. The master SPI must be configured for the MODF feature (see Table 21-3). After MODF is set, the current transfer is aborted and the following bit is changed:

- MSTR = 0, The master bit in SPICR1 resets.

The MODF interrupt is reflected in the status register MODF flag. Clearing the flag will also clear the interrupt. This interrupt will stay active while the MODF flag is set. MODF has an automatic clearing process which is described in Section 21.3.2.4, “SPI Status Register (SPISR)”.

#### 21.4.7.5.2 SPIF

SPIF occurs when new data has been received and copied to the SPI data register. After SPIF is set, it does not clear until it is serviced. SPIF has an automatic clearing process, which is described in Section 21.3.2.4, “SPI Status Register (SPISR)”.

#### 21.4.7.5.3 SPTEF

SPTEF occurs when the SPI data register is ready to accept new data. After SPTEF is set, it does not clear until it is serviced. SPTEF has an automatic clearing process, which is described in Section 21.3.2.4, “SPI Status Register (SPISR)”.

# Chapter 22
# Timer Module (TIM16B8CV2) Block Description

**Table 22-1. Revision History**

| Revision Number | Revision Date | Sections Affected | Description of Changes |
|---|---|---|---|
| V02.05 | 9 Jul 2009 | 22.3.2.12/22-803<br>22.3.2.13/22-803<br>22.3.2.15/22-805<br>22.3.2.16/22-806<br>22.3.2.19/22-808<br>22.4.2/22-811<br>22.4.3/22-811 | - Revised flag clearing procedure, whereby TEN or PAEN bit must be set when clearing flags.<br>- Add fomula to describe prescaler |
| V02.06 | 26 Aug 2009 | 22.1.2/22-788<br>22.3.2.15/22-805<br>22.3.2.2/22-794<br>22.3.2.3/22-795<br>22.3.2.4/22-796<br>22.4.3/22-811 | - Correct typo: TSCR ->TSCR1<br>- Correct reference: Figure 1-25 -> Figure 1-31<br>- Add description, “a counter overflow when TTOV[7] is set”, to be the condition of channel 7 override event.<br>- Phrase the description of OC7M to make it more explicit |
| V02.07 | 04 May 2010 | 22.3.2.8/22-799<br>22.3.2.11/22-802<br>22.4.3/22-811 | - Add Table 22-10<br>- in TCRE bit description part,add Note<br>- Add Figure 22-31 |

## 22.1 Introduction

The basic timer consists of a 16-bit, software-programmable counter driven by a enhanced programmable prescaler.

This timer can be used for many purposes, including input waveform measurements while simultaneously generating an output waveform. Pulse widths can vary from microseconds to many seconds.

This timer contains 8 complete input capture/output compare channels and one pulse accumulator. The input capture function is used to detect a selected transition edge and record the time. The output compare function is used for generating output signals or for timer software delays. The 16-bit pulse accumulator is used to operate as a simple event counter or a gated time accumulator. The pulse accumulator shares timer channel 7 when in event mode.

A full access for the counter registers or the input capture/output compare registers should take place in one clock cycle. Accessing high byte and low byte separately for all of these registers may not yield the same result as accessing them in one word.

### 22.1.1 Features

The TIM16B8CV2 includes these distinctive features:

- Eight input capture/output compare channels.
- Clock prescaling.
- 16-bit counter.
- 16-bit pulse accumulator.

### 22.1.2 Modes of Operation

Stop: Timer is off because clocks are stopped.

Freeze: Timer counter keep on running, unless TSFRZ in TSCR1 (0x0006) is set to 1.

Wait: Counters keep on running, unless TSWAI in TSCR1 (0x0006) is set to 1.

Normal: Timer counter keep on running, unless TEN in TSCR1 (0x0006) is cleared to 0.

### 22.1.3 Block Diagrams

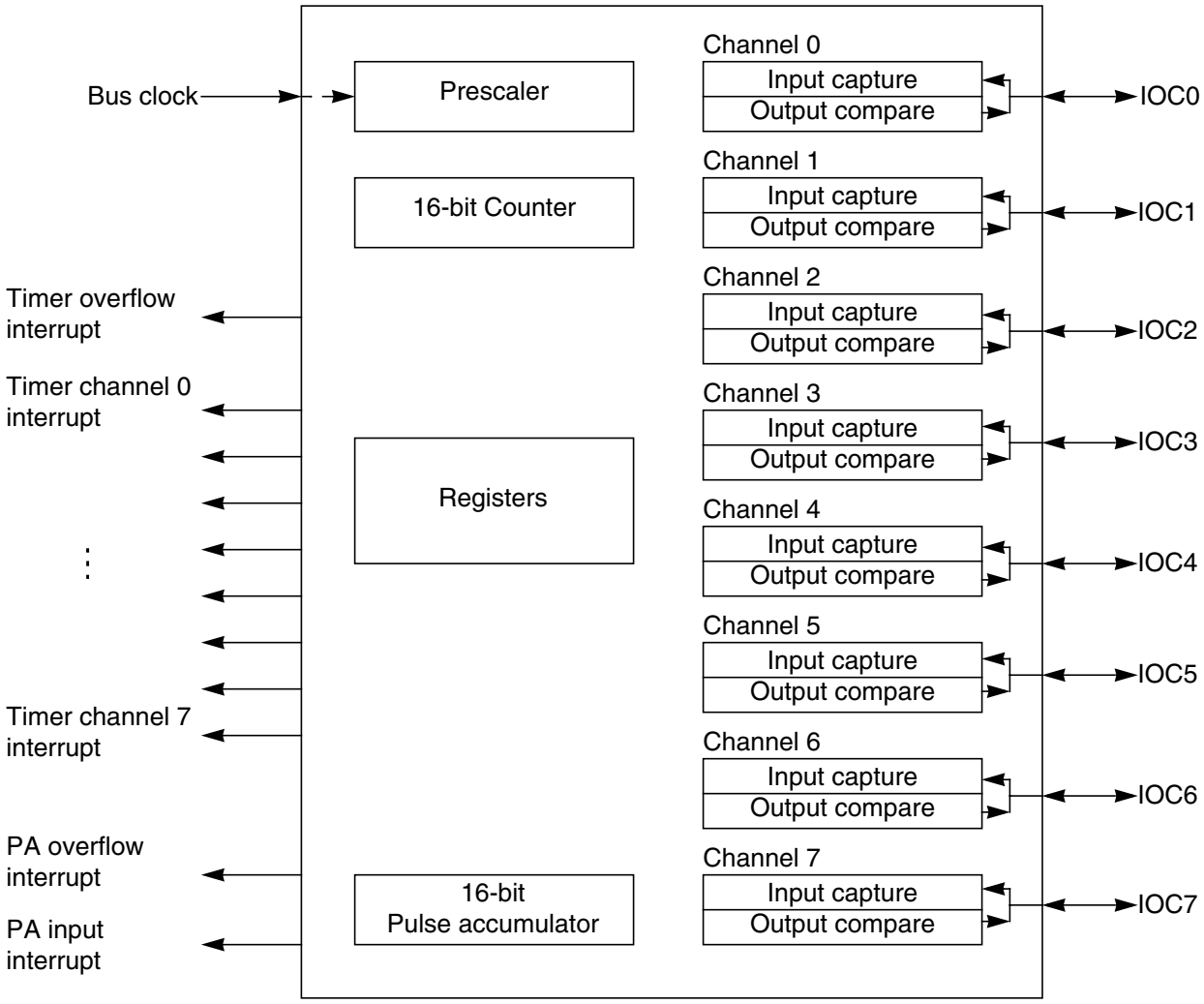



**Figure 22-1. TIM16B8CV2 Block Diagram**

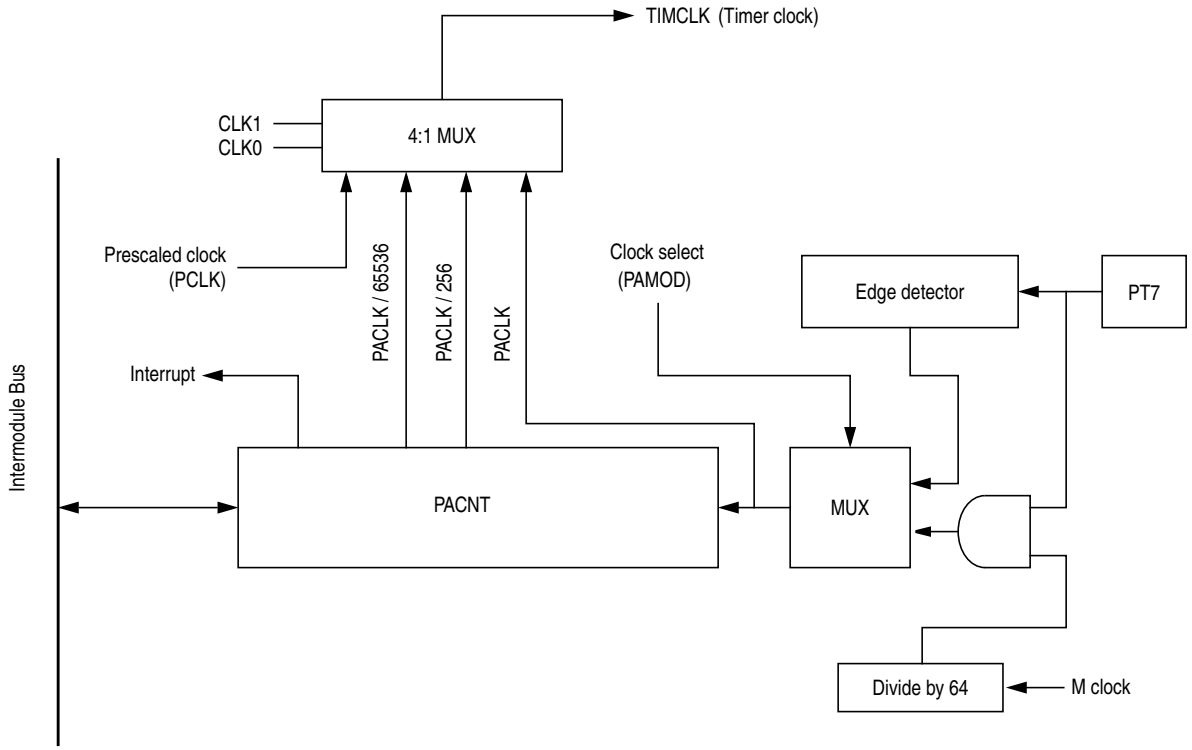



**Figure 22-2. 16-Bit Pulse Accumulator Block Diagram**

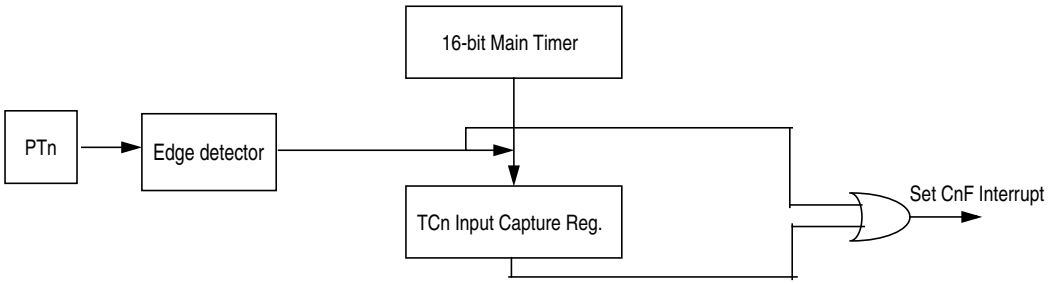



**Figure 22-3. Interrupt Flag Setting**

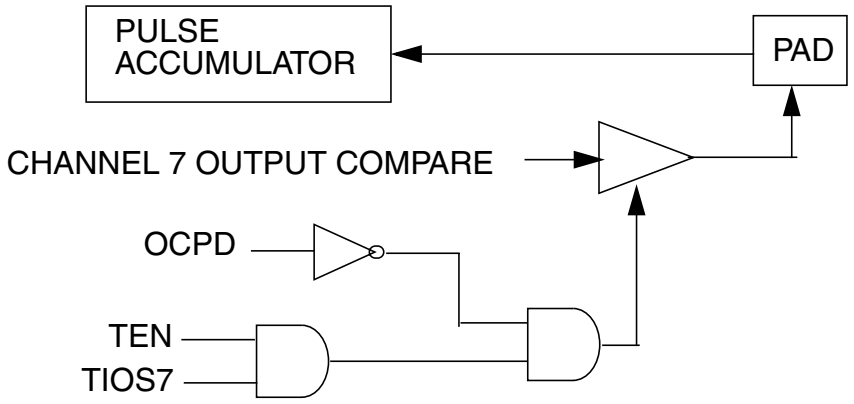



**Figure 22-4. Channel 7 Output Compare/Pulse Accumulator Logic**

## 22.2 External Signal Description

The TIM16B8CV2 module has a total of eight external pins.

### 22.2.1 IOC7 — Input Capture and Output Compare Channel 7 Pin

This pin serves as input capture or output compare for channel 7. This can also be configured as pulse accumulator input.

### 22.2.2 IOC6 — Input Capture and Output Compare Channel 6 Pin

This pin serves as input capture or output compare for channel 6.

### 22.2.3 IOC5 — Input Capture and Output Compare Channel 5 Pin

This pin serves as input capture or output compare for channel 5.

### 22.2.4 IOC4 — Input Capture and Output Compare Channel 4 Pin

This pin serves as input capture or output compare for channel 4. Pin

### 22.2.5 IOC3 — Input Capture and Output Compare Channel 3 Pin

This pin serves as input capture or output compare for channel 3.

### 22.2.6 IOC2 — Input Capture and Output Compare Channel 2 Pin

This pin serves as input capture or output compare for channel 2.

### 22.2.7 IOC1 — Input Capture and Output Compare Channel 1 Pin

This pin serves as input capture or output compare for channel 1.

### 22.2.8 IOC0 — Input Capture and Output Compare Channel 0 Pin

This pin serves as input capture or output compare for channel 0.

**NOTE**

For the description of interrupts see Section 22.6, “Interrupts”.

## 22.3 Memory Map and Register Definition

This section provides a detailed description of all memory and registers.

### 22.3.1 Module Memory Map

The memory map for the TIM16B8CV2 module is given below in Figure 22-5. The address listed for each register is the address offset. The total address for each register is the sum of the base address for the TIM16B8CV2 module and the address offset for each register.

### 22.3.2 Register Descriptions

This section consists of register descriptions in address order. Each description includes a standard register diagram with an associated figure number. Details of register bit and field function follow the register diagrams, in bit order.

| Register Name | | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|---|
| 0x0000 TIOS | R<br>W | IOS7 | IOS6 | IOS5 | IOS4 | IOS3 | IOS2 | IOS1 | IOS0 |
| 0x0001 CFORC | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | W | FOC7 | FOC6 | FOC5 | FOC4 | FOC3 | FOC2 | FOC1 | FOC0 |
| 0x0002 OC7M | R<br>W | OC7M7 | OC7M6 | OC7M5 | OC7M4 | OC7M3 | OC7M2 | OC7M1 | OC7M0 |
| 0x0003 OC7D | R<br>W | OC7D7 | OC7D6 | OC7D5 | OC7D4 | OC7D3 | OC7D2 | OC7D1 | OC7D0 |
| 0x0004 TCNTH | R<br>W | TCNT15 | TCNT14 | TCNT13 | TCNT12 | TCNT11 | TCNT10 | TCNT9 | TCNT8 |
| 0x0005 TCNTL | R<br>W | TCNT7 | TCNT6 | TCNT5 | TCNT4 | TCNT3 | TCNT2 | TCNT1 | TCNT0 |

= Unimplemented or Reserved

**Figure 22-5. TIM16B8CV2 Register Summary (Sheet 1 of 3)**

| Register Name | | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|---|
| 0x0006 TSCR1 | R | TEN | TSWAI | TSFRZ | TFFCA | PRNT | 0 | 0 | 0 |
| | W | | | | | | | | |
| 0x0007 TTOV | R | TOV7 | TOV6 | TOV5 | TOV4 | TOV3 | TOV2 | TOV1 | TOV0 |
| | W | | | | | | | | |
| 0x0008 TCTL1 | R | OM7 | OL7 | OM6 | OL6 | OM5 | OL5 | OM4 | OL4 |
| | W | | | | | | | | |
| 0x0009 TCTL2 | R | OM3 | OL3 | OM2 | OL2 | OM1 | OL1 | OM0 | OL0 |
| | W | | | | | | | | |
| 0x000A TCTL3 | R | EDG7B | EDG7A | EDG6B | EDG6A | EDG5B | EDG5A | EDG4B | EDG4A |
| | W | | | | | | | | |
| 0x000B TCTL4 | R | EDG3B | EDG3A | EDG2B | EDG2A | EDG1B | EDG1A | EDG0B | EDG0A |
| | W | | | | | | | | |
| 0x000C TIE | R | C7I | C6I | C5I | C4I | C3I | C2I | C1I | C0I |
| | W | | | | | | | | |
| 0x000D TSCR2 | R | TOI | 0 | 0 | 0 | TCRE | PR2 | PR1 | PR0 |
| | W | | | | | | | | |
| 0x000E TFLG1 | R | C7F | C6F | C5F | C4F | C3F | C2F | C1F | C0F |
| | W | | | | | | | | |
| 0x000F TFLG2 | R | TOF | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | W | | | | | | | | |
| 0x0010–0x001F TCxH–TCxL | R | Bit 15 | Bit 14 | Bit 13 | Bit 12 | Bit 11 | Bit 10 | Bit 9 | Bit 8 |
| | W | | | | | | | | |
| | R | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
| | W | | | | | | | | |
| 0x0020 PACTL | R | 0 | PAEN | PAMOD | PEDGE | CLK1 | CLK0 | PAOVI | PAI |
| | W | | | | | | | | |
| 0x0021 PAFLG | R | 0 | 0 | 0 | 0 | 0 | 0 | PAOVF | PAIF |
| | W | | | | | | | | |
| 0x0022 PACNTH | R | PACNT15 | PACNT14 | PACNT13 | PACNT12 | PACNT11 | PACNT10 | PACNT9 | PACNT8 |
| | W | | | | | | | | |
| 0x0023 PACNTL | R | PACNT7 | PACNT6 | PACNT5 | PACNT4 | PACNT3 | PACNT2 | PACNT1 | PACNT0 |
| | W | | | | | | | | |
| 0x0024–0x002B Reserved | R | | | | | | | | |
| | W | | | | | | | | |

= Unimplemented or Reserved

**Figure 22-5. TIM16B8CV2 Register Summary (Sheet 2 of 3)**

| Register Name | | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|---|
| 0x002C OCPD | R<br>W | OCPD7 | OCPD6 | OCPD5 | OCPD4 | OCPD3 | OCPD2 | OCPD1 | OCPD0 |
| 0x002D | R | | | | | | | | |
| 0x002E PTPSR | R<br>W | PTPS7 | PTPS6 | PTPS5 | PTPS4 | PTPS3 | PTPS2 | PTPS1 | PTPS0 |
| 0x002F Reserved | R<br>W | | | | | | | | |

= Unimplemented or Reserved

**Figure 22-5. TIM16B8CV2 Register Summary (Sheet 3 of 3)**

### 22.3.2.1 Timer Input Capture/Output Compare Select (TIOS)

Module Base + 0x0000

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | IOS7 | IOS6 | IOS5 | IOS4 | IOS3 | IOS2 | IOS1 | IOS0 |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 22-6. Timer Input Capture/Output Compare Select (TIOS)**

Read: Anytime

Write: Anytime

**Table 22-2. TIOS Field Descriptions**

| Field | Description |
|---|---|
| 7:0<br>IOS[7:0] | **Input Capture or Output Compare Channel Configuration**<br>0 The corresponding channel acts as an input capture.<br>1 The corresponding channel acts as an output compare. |

### 22.3.2.2 Timer Compare Force Register (CFORC)

Module Base + 0x0001

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| W | FOC7 | FOC6 | FOC5 | FOC4 | FOC3 | FOC2 | FOC1 | FOC0 |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 22-7. Timer Compare Force Register (CFORC)**

Read: Anytime but will always return 0x0000 (1 state is transient)

Write: Anytime

**Table 22-3. CFORC Field Descriptions**

| Field | Description |
|---|---|
| 7:0<br>FOC[7:0] | **Force Output Compare Action for Channel 7:0** — A write to this register with the corresponding data bit(s) set causes the action which is programmed for output compare “x” to occur immediately. The action taken is the same as if a successful comparison had just taken place with the TCx register except the interrupt flag does not get set.<br>**Note:** A channel 7 event, which can be a counter overflow when TTOV[7] is set or a successful output compare on channel 7, overrides any channel 6:0 compares. If forced output compare on any channel occurs at the same time as the successful output compare then forced output compare action will take precedence and interrupt flag won’t get set. |

### 22.3.2.3 Output Compare 7 Mask Register (OC7M)

Module Base + 0x0002

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | OC7M7 | OC7M6 | OC7M5 | OC7M4 | OC7M3 | OC7M2 | OC7M1 | OC7M0 |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 22-8. Output Compare 7 Mask Register (OC7M)**

Read: Anytime

Write: Anytime

**Table 22-4. OC7M Field Descriptions**

| Field | Description |
|---|---|
| 7:0<br>OC7M[7:0] | **Output Compare 7 Mask** — A channel 7 event, which can be a counter overflow when TTOV[7] is set or a successful output compare on channel 7, overrides any channel 6:0 compares. For each OC7M bit that is set, the output compare action reflects the corresponding OC7D bit.<br>0 The corresponding OC7Dx bit in the output compare 7 data register will not be transferred to the timer port on a channel 7 event, even if the corresponding pin is setup for output compare.<br>1 The corresponding OC7Dx bit in the output compare 7 data register will be transferred to the timer port on a channel 7 event.<br>**Note:** The corresponding channel must also be setup for output compare (IOSx = 1 and OCPDx = 0) for data to be transferred from the output compare 7 data register to the timer port. |

### 22.3.2.4 Output Compare 7 Data Register (OC7D)

Module Base + 0x0003

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | OC7D7 | OC7D6 | OC7D5 | OC7D4 | OC7D3 | OC7D2 | OC7D1 | OC7D0 |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 22-9. Output Compare 7 Data Register (OC7D)**

Read: Anytime

Write: Anytime

**Table 22-5. OC7D Field Descriptions**

| Field | Description |
|---|---|
| 7:0<br>OC7D[7:0] | **Output Compare 7 Data** — A channel 7 event, which can be a counter overflow when TTOV[7] is set or a successful output compare on channel 7, can cause bits in the output compare 7 data register to transfer to the timer port data register depending on the output compare 7 mask register. |

### 22.3.2.5 Timer Count Register (TCNT)

Module Base + 0x0004

| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 9 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | TCNT15 | TCNT14 | TCNT13 | TCNT12 | TCNT11 | TCNT10 | TCNT9 | TCNT8 |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 22-10. Timer Count Register High (TCNTH)**

Module Base + 0x0005

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | TCNT7 | TCNT6 | TCNT5 | TCNT4 | TCNT3 | TCNT2 | TCNT1 | TCNT0 |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 22-11. Timer Count Register Low (TCNTL)**

The 16-bit main timer is an up counter.

A full access for the counter register should take place in one clock cycle. A separate read/write for high byte and low byte will give a different result than accessing them as a word.

Read: Anytime

Write: Has no meaning or effect in the normal mode; only writable in special modes (test_mode = 1).

The period of the first count after a write to the TCNT registers may be a different size because the write is not synchronized with the prescaler clock.

### 22.3.2.6 Timer System Control Register 1 (TSCR1)

Module Base + 0x0006

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | TEN | TSWAI | TSFRZ | TFFCA | PRNT | 0 | 0 | 0 |
| W | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented or Reserved

**Figure 22-12. Timer System Control Register 1 (TSCR1)**

Read: Anytime

Write: Anytime

**Table 22-6. TSCR1 Field Descriptions**

| Field | Description |
|---|---|
| 7<br>TEN | **Timer Enable**<br>0 Disables the main timer, including the counter. Can be used for reducing power consumption.<br>1 Allows the timer to function normally.<br>If for any reason the timer is not active, there is no ÷64 clock for the pulse accumulator because the ÷64 is generated by the timer prescaler. |
| 6<br>TSWAI | **Timer Module Stops While in Wait**<br>0 Allows the timer module to continue running during wait.<br>1 Disables the timer module when the MCU is in the wait mode. Timer interrupts cannot be used to get the MCU out of wait.<br>TSWAI also affects pulse accumulator. |
| 5<br>TSFRZ | **Timer Stops While in Freeze Mode**<br>0 Allows the timer counter to continue running while in freeze mode.<br>1 Disables the timer counter whenever the MCU is in freeze mode. This is useful for emulation.<br>TSFRZ does not stop the pulse accumulator. |

**Table 22-6. TSCR1 Field Descriptions (continued)**

| Field | Description |
|---|---|
| 4<br>TFFCA | **Timer Fast Flag Clear All**<br>0 Allows the timer flag clearing to function normally.<br>1 For TFLG1(0x000E), a read from an input capture or a write to the output compare channel (0x0010–0x001F) causes the corresponding channel flag, CnF, to be cleared. For TFLG2 (0x000F), any access to the TCNT register (0x0004, 0x0005) clears the TOF flag. Any access to the PACNT registers (0x0022, 0x0023) clears the PAOVF and PAIF flags in the PAFLG register (0x0021). This has the advantage of eliminating software overhead in a separate clear sequence. Extra care is required to avoid accidental flag clearing due to unintended accesses. |
| 3<br>PRNT | **Precision Timer**<br>0 Enables legacy timer. PR0, PR1, and PR2 bits of the TSCR2 register are used for timer counter prescaler selection.<br>1 Enables precision timer. All bits of the PTPSR register are used for Precision Timer Prescaler Selection, and all bits.<br>This bit is writable only once out of reset. |

### 22.3.2.7 Timer Toggle On Overflow Register 1 (TTOV)

Module Base + 0x0007

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | TOV7 | TOV6 | TOV5 | TOV4 | TOV3 | TOV2 | TOV1 | TOV0 |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 22-13. Timer Toggle On Overflow Register 1 (TTOV)**

Read: Anytime

Write: Anytime

**Table 22-7. TTOV Field Descriptions**

| Field | Description |
|---|---|
| 7:0<br>TOV[7:0] | **Toggle On Overflow Bits** — TOVx toggles output compare pin on overflow. This feature only takes effect when in output compare mode. When set, it takes precedence over forced output compare but not channel 7 override events.<br>0 Toggle output compare pin on overflow feature disabled.<br>1 Toggle output compare pin on overflow feature enabled. |

### 22.3.2.8 Timer Control Register 1/Timer Control Register 2 (TCTL1/TCTL2)

Module Base + 0x0008

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | OM7 | OL7 | OM6 | OL6 | OM5 | OL5 | OM4 | OL4 |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 22-14. Timer Control Register 1 (TCTL1)**

Module Base + 0x0009

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | OM3 | OL3 | OM2 | OL2 | OM1 | OL1 | OM0 | OL0 |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 22-15. Timer Control Register 2 (TCTL2)**

Read: Anytime

Write: Anytime

**Table 22-8. TCTL1/TCTL2 Field Descriptions**

| Field | Description |
|---|---|
| 7:0<br>OMx | **Output Mode** — These eight pairs of control bits are encoded to specify the output action to be taken as a result of a successful OCx compare. When either OMx or OLx is 1, the pin associated with OCx becomes an output tied to OCx.<br>**Note:** To enable output action by OMx bits on timer port, the corresponding bit in OC7M should be cleared. For an output line to be driven by an OCx the OCPDx must be cleared. |
| 7:0<br>OLx | **Output Level** — These eight pairs of control bits are encoded to specify the output action to be taken as a result of a successful OCx compare. When either OMx or OLx is 1, the pin associated with OCx becomes an output tied to OCx.<br>**Note:** To enable output action by OLx bits on timer port, the corresponding bit in OC7M should be cleared. For an output line to be driven by an OCx the OCPDx must be cleared. |

**Table 22-9. Compare Result Output Action**

| OMx | OLx | Action |
|---|---|---|
| 0 | 0 | No output compare<br>action on the timer output signal |
| 0 | 1 | Toggle OCx output line |
| 1 | 0 | Clear OCx output line to zero |
| 1 | 1 | Set OCx output line to one |

To operate the 16-bit pulse accumulator independently of input capture or output compare 7 and 0 respectively the user must set the corresponding bits IOSx = 1, OMx = 0 and OLx = 0. OC7M7 in the OC7M register must also be cleared.

To enable output action using the OM7 and OL7 bits on the timer port,the corresponding bit OC7M7 in the OC7M register must also be cleared. The settings for these bits can be seen in Table 22-10

**Table 22-10. The OC7 and OCx event priority**

| OC7M7=0 | | | | OC7M7=1 | | | |
|---|---|---|---|---|---|---|---|
| OC7Mx=1 | | OC7Mx=0 | | OC7Mx=1 | | OC7Mx=0 | |
| TC7=TCx | TC7>TCx | TC7=TCx | TC7>TCx | TC7=TCx | TC7>TCx | TC7=TCx | TC7>TCx |
| IOCx=OC7Dx IOC7=OM7/OL7 | IOCx=OC7Dx +OMx/OLx IOC7=OM7/OL7 | IOCx=OMx/OLx IOC7=OM7/OL7 | | IOCx=OC7Dx IOC7=OC7D7 | IOCx=OC7Dx +OMx/OLx IOC7=OC7D7 | IOCx=OMx/OLx IOC7=OC7D7 | |

Note: in Table 22-10, the IOS7 and IOSx should be set to 1

IOSx is the register TIOS bit x,

OC7Mx is the register OC7M bit x,

TCx is timer Input Capture/Output Compare register,

IOCx is channel x,

OMx/OLx is the register TCTL1/TCTL2,

OC7Dx is the register OC7D bit x.

IOCx = OC7Dx+ OMx/OLx, means that both OC7 event and OCx event will change channel x value.

### 22.3.2.9 Timer Control Register 3/Timer Control Register 4 (TCTL3 and TCTL4)

Module Base + 0x000A

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | EDG7B | EDG7A | EDG6B | EDG6A | EDG5B | EDG5A | EDG4B | EDG4A |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 22-16. Timer Control Register 3 (TCTL3)**

Module Base + 0x000B

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | EDG3B | EDG3A | EDG2B | EDG2A | EDG1B | EDG1A | EDG0B | EDG0A |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 22-17. Timer Control Register 4 (TCTL4)**

Read: Anytime

Write: Anytime.

**Table 22-11. TCTL3/TCTL4 Field Descriptions**

| Field | Description |
|---|---|
| 7:0<br>EDGnB<br>EDGnA | **Input Capture Edge Control** — These eight pairs of control bits configure the input capture edge detector circuits. |

**Table 22-12. Edge Detector Circuit Configuration**

| EDGnB | EDGnA | Configuration |
|---|---|---|
| 0 | 0 | Capture disabled |
| 0 | 1 | Capture on rising edges only |
| 1 | 0 | Capture on falling edges only |
| 1 | 1 | Capture on any edge (rising or falling) |

### 22.3.2.10 Timer Interrupt Enable Register (TIE)

Module Base + 0x000C

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | C7I | C6I | C5I | C4I | C3I | C2I | C1I | C0I |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 22-18. Timer Interrupt Enable Register (TIE)**

Read: Anytime

Write: Anytime.

**Table 22-13. TIE Field Descriptions**

| Field | Description |
|---|---|
| 7:0<br>C7I:C0I | **Input Capture/Output Compare “x” Interrupt Enable —** The bits in TIE correspond bit-for-bit with the bits in the TFLG1 status register. If cleared, the corresponding flag is disabled from causing a hardware interrupt. If set, the corresponding flag is enabled to cause a interrupt. |

### 22.3.2.11 Timer System Control Register 2 (TSCR2)

Module Base + 0x000D

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | TOI | 0 | 0 | 0 | TCRE | PR2 | PR1 | PR0 |
| W | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented or Reserved

**Figure 22-19. Timer System Control Register 2 (TSCR2)**

Read: Anytime

Write: Anytime.

**Table 22-14. TSCR2 Field Descriptions**

| Field | Description |
|---|---|
| 7<br>TOI | **Timer Overflow Interrupt Enable**<br>0 Interrupt inhibited.<br>1 Hardware interrupt requested when TOF flag set. |
| 3<br>TCRE | **Timer Counter Reset Enable** — This bit allows the timer counter to be reset by a successful output compare 7 event. This mode of operation is similar to an up-counting modulus counter.<br>0 Counter reset inhibited and counter free runs.<br>1 Counter reset by a successful output compare 7.<br>**Note:** If TC7 = 0x0000 and TCRE = 1, TCNT will stay at 0x0000 continuously. If TC7 = 0xFFFF and TCRE = 1, TOF will never be set when TCNT is reset from 0xFFFF to 0x0000.<br>**Note:** TCRE=1 and TC7!=0, the TCNT cycle period will be TC7 x "prescaler counter width" + "1 Bus Clock", for a more detail explanation please refer to Section 22.4.3, “Output Compare |
| 2<br>PR[2:0] | **Timer Prescaler Select** — These three bits select the frequency of the timer prescaler clock derived from the Bus Clock as shown in Table 22-15. |

**Table 22-15. Timer Clock Selection**

| PR2 | PR1 | PR0 | Timer Clock |
|---|---|---|---|
| 0 | 0 | 0 | Bus Clock / 1 |
| 0 | 0 | 1 | Bus Clock / 2 |
| 0 | 1 | 0 | Bus Clock / 4 |
| 0 | 1 | 1 | Bus Clock / 8 |
| 1 | 0 | 0 | Bus Clock / 16 |
| 1 | 0 | 1 | Bus Clock / 32 |
| 1 | 1 | 0 | Bus Clock / 64 |
| 1 | 1 | 1 | Bus Clock / 128 |

**NOTE**

The newly selected prescale factor will not take effect until the next synchronized edge where all prescale counter stages equal zero.

### 22.3.2.12 Main Timer Interrupt Flag 1 (TFLG1)

Module Base + 0x000E

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | C7F | C6F | C5F | C4F | C3F | C2F | C1F | C0F |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 22-20. Main Timer Interrupt Flag 1 (TFLG1)**

Read: Anytime

Write: Used in the clearing mechanism (set bits cause corresponding bits to be cleared). Writing a zero will not affect current status of the bit.

**Table 22-16. TRLG1 Field Descriptions**

| Field | Description |
|---|---|
| 7:0<br>C[7:0]F | **Input Capture/Output Compare Channel "x" Flag** — These flags are set when an input capture or output compare event occurs. Clearing requires writing a one to the corresponding flag bit while TEN or PAEN is set to one.<br><br>When TFFCA bit in TSCR register is set, a read from an input capture or a write into an output compare channel (0x0010–0x001F) will cause the corresponding channel flag CxF to be cleared. |

### 22.3.2.13 Main Timer Interrupt Flag 2 (TFLG2)

Module Base + 0x000F

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | TOF | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| W | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

Unimplemented or Reserved

**Figure 22-21. Main Timer Interrupt Flag 2 (TFLG2)**

TFLG2 indicates when interrupt conditions have occurred. To clear a bit in the flag register, write the bit to one while TEN bit of TSCR1 or PAEN bit of PACTL is set to one.

Read: Anytime

Write: Used in clearing mechanism (set bits cause corresponding bits to be cleared).

Any access to TCNT will clear TFLG2 register if the TFFCA bit in TSCR register is set.

**Table 22-17. TRLG2 Field Descriptions**

| Field | Description |
|---|---|
| 7<br>TOF | **Timer Overflow Flag** — Set when 16-bit free-running timer overflows from 0xFFFF to 0x0000. Clearing this bit requires writing a one to bit 7 of TFLG2 register while the TEN bit of TSCR1 or PAEN bit of PACTL is set to one (See also TCRE control bit explanation.) |

### 22.3.2.14 Timer Input Capture/Output Compare Registers High and Low 0–7 (TCxH and TCxL)

Module Base + 0x0010 = TC0H, 0x0012 = TC1H, 0x0014 = TC2H, 0x0016 = TC3H, 0x0018 = TC4H, 0x001A = TC5H, 0x001C = TC6H, 0x001E = TC7H

| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 0 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | Bit 15 | Bit 14 | Bit 13 | Bit 12 | Bit 11 | Bit 10 | Bit 9 | Bit 8 |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 22-22. Timer Input Capture/Output Compare Register x High (TCxH)**

Module Base + 0x0011 = TC0L, 0x0013 = TC1L, 0x0015 = TC2L, 0x0017 = TC3L, 0x0019 = TC4L, 0x001B = TC5L, 0x001D = TC6L, 0x001F = TC7L

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 22-23. Timer Input Capture/Output Compare Register x Low (TCxL)**

Depending on the TIOS bit for the corresponding channel, these registers are used to latch the value of the free-running counter when a defined transition is sensed by the corresponding input capture edge detector or to trigger an output action for output compare.

Read: Anytime

Write: Anytime for output compare function.Writes to these registers have no meaning or effect during input capture. All timer input capture/output compare registers are reset to 0x0000.

**NOTE**

Read/Write access in byte mode for high byte should takes place before low byte otherwise it will give a different result.

### 22.3.2.15 16-Bit Pulse Accumulator Control Register (PACTL)

Module Base + 0x0020

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | 0 | PAEN | PAMOD | PEDGE | CLK1 | CLK0 | PAOVI | PAI |
| W | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

Unimplemented or Reserved

**Figure 22-24. 16-Bit Pulse Accumulator Control Register (PACTL)**

When PAEN is set, the PACT is enabled.The PACT shares the input pin with IOC7.

Read: Any time

Write: Any time

**Table 22-18. PACTL Field Descriptions**

| Field | Description |
|---|---|
| 6<br>PAEN | **Pulse Accumulator System Enable** — PAEN is independent from TEN. With timer disabled, the pulse accumulator can function unless pulse accumulator is disabled.<br>0 16-Bit Pulse Accumulator system disabled.<br>1 Pulse Accumulator system enabled. |
| 5<br>PAMOD | **Pulse Accumulator Mode** — This bit is active only when the Pulse Accumulator is enabled (PAEN = 1). See Table 22-19.<br>0 Event counter mode.<br>1 Gated time accumulation mode. |
| 4<br>PEDGE | **Pulse Accumulator Edge Control** — This bit is active only when the Pulse Accumulator is enabled (PAEN = 1). For PAMOD bit = 0 (event counter mode). See Table 22-19.<br>0 Falling edges on IOC7 pin cause the count to be incremented.<br>1 Rising edges on IOC7 pin cause the count to be incremented.<br>For PAMOD bit = 1 (gated time accumulation mode).<br>0 IOC7 input pin high enables M (bus clock) divided by 64 clock to Pulse Accumulator and the trailing falling edge on IOC7 sets the PAIF flag.<br>1 IOC7 input pin low enables M (bus clock) divided by 64 clock to Pulse Accumulator and the trailing rising edge on IOC7 sets the PAIF flag. |
| 3:2<br>CLK[1:0] | **Clock Select Bits —** Refer to Table 22-20. |
| 1<br>PAOVI | **Pulse Accumulator Overflow Interrupt Enable**<br>0 Interrupt inhibited.<br>1 Interrupt requested if PAOVF is set. |
| 0<br>PAI | **Pulse Accumulator Input Interrupt Enable**<br>0 Interrupt inhibited.<br>1 Interrupt requested if PAIF is set. |

**Table 22-19. Pin Action**

| PAMOD | PEDGE | Pin Action |
|---|---|---|
| 0 | 0 | Falling edge |
| 0 | 1 | Rising edge |
| 1 | 0 | Div. by 64 clock enabled with pin high level |
| 1 | 1 | Div. by 64 clock enabled with pin low level |

**NOTE**

If the timer is not active (TEN = 0 in TSCR), there is no divide-by-64 because the ÷64 clock is generated by the timer prescaler.

**Table 22-20. Timer Clock Selection**

| CLK1 | CLK0 | Timer Clock |
|---|---|---|
| 0 | 0 | Use timer prescaler clock as timer counter clock |
| 0 | 1 | Use PACLK as input to timer counter clock |
| 1 | 0 | Use PACLK/256 as timer counter clock frequency |
| 1 | 1 | Use PACLK/65536 as timer counter clock frequency |

For the description of PACLK please refer Figure 22-30**.**

If the pulse accumulator is disabled (PAEN = 0), the prescaler clock from the timer is always used as an input clock to the timer counter. The change from one selected clock to the other happens immediately after these bits are written.

### 22.3.2.16 Pulse Accumulator Flag Register (PAFLG)

Module Base + 0x0021

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | 0 | 0 | 0 | 0 | PAOVF | PAIF |
| W | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

Unimplemented or Reserved

**Figure 22-25. Pulse Accumulator Flag Register (PAFLG)**

Read: Anytime

Write: Anytime

When the TFFCA bit in the TSCR register is set, any access to the PACNT register will clear all the flags in the PAFLG register. Timer module or Pulse Accumulator must stay enabled (TEN=1 or PAEN=1) while clearing these bits.

**Table 22-21. PAFLG Field Descriptions**

| Field | Description |
|---|---|
| 1<br>PAOVF | **Pulse Accumulator Overflow Flag** — Set when the 16-bit pulse accumulator overflows from 0xFFFF to 0x0000.<br>Clearing this bit requires writing a one to this bit in the PAFLG register while TEN bit of TSCR1 or PAEN bit of PACTL register is set to one. |
| 0<br>PAIF | **Pulse Accumulator Input edge Flag** — Set when the selected edge is detected at the IOC7 input pin.In event mode the event edge triggers PAIF and in gated time accumulation mode the trailing edge of the gate signal at the IOC7 input pin triggers PAIF.<br>Clearing this bit requires writing a one to this bit in the PAFLG register while TEN bit of TSCR1 or PAEN bit of PACTL register is set to one. Any access to the PACNT register will clear all the flags in this register when TFFCA bit in register TSCR(0x0006) is set. |

## 22.3.2.17 Pulse Accumulators Count Registers (PACNT)

Module Base + 0x0022

| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 0 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | PACNT15 | PACNT14 | PACNT13 | PACNT12 | PACNT11 | PACNT10 | PACNT9 | PACNT8 |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 22-26. Pulse Accumulator Count Register High (PACNTH)**

Module Base + 0x0023

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | PACNT7 | PACNT6 | PACNT5 | PACNT4 | PACNT3 | PACNT2 | PACNT1 | PACNT0 |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 22-27. Pulse Accumulator Count Register Low (PACNTL)**

Read: Anytime

Write: Anytime

These registers contain the number of active input edges on its input pin since the last reset.

When PACNT overflows from 0xFFFF to 0x0000, the Interrupt flag PAOVF in PAFLG (0x0021) is set.

Full count register access should take place in one clock cycle. A separate read/write for high byte and low byte will give a different result than accessing them as a word.

**NOTE**

Reading the pulse accumulator counter registers immediately after an active edge on the pulse accumulator input pin may miss the last count because the input has to be synchronized with the bus clock first.

### 22.3.2.18 Output Compare Pin Disconnect Register(OCPD)

Module Base + 0x002C

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | OCPD7 | OCPD6 | OCPD5 | OCPD4 | OCPD3 | OCPD2 | OCPD1 | OCPD0 |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 22-28. Ouput Compare Pin Disconnect Register (OCPD)**

Read: Anytime

Write: Anytime

All bits reset to zero.

**Table 22-22. OCPD Field Description**

| Field | Description |
|---|---|
| OCPD[7:0} | **Output Compare Pin Disconnect Bits**<br>0 Enables the timer channel port. Ouptut Compare action will occur on the channel pin. These bits do not affect the input capture or pulse accumulator functions<br>1 Disables the timer channel port. Output Compare action will not occur on the channel pin, but the output compare flag still become set . |

### 22.3.2.19 Precision Timer Prescaler Select Register (PTPSR)

Module Base + 0x002E

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | PTPS7 | PTPS6 | PTPS5 | PTPS4 | PTPS3 | PTPS2 | PTPS1 | PTPS0 |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 22-29. Precision Timer Prescaler Select Register (PTPSR)**

Read: Anytime

Write: Anytime

All bits reset to zero.

**Table 22-23. PTPSR Field Descriptions**

| Field | Description |
|---|---|
| 7:0<br>PTPS[7:0] | **Precision Timer Prescaler Select Bits** — These eight bits specify the division rate of the main Timer prescaler. These are effective only when the PRNT bit of TSCR1 is set to 1. Table 22-24 shows some selection examples in this case.<br>The newly selected prescale factor will not take effect until the next synchronized edge where all prescale counter stages equal zero. |

The Prescaler can be calculated as follows depending on logical value of the PTPS[7:0] and PRNT bit:

PRNT = 1 : Prescaler = PTPS[7:0] + 1

**Table 22-24. Precision Timer Prescaler Selection Examples when PRNT = 1**

| PTPS7 | PTPS6 | PTPS5 | PTPS4 | PTPS3 | PTPS2 | PTPS1 | PTPS0 | Prescale Factor |
|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 2 |
| 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 3 |
| 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 4 |
| 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 5 |
| 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 6 |
| 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 7 |
| 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 8 |
| 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 16 |
| 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 32 |
| 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 64 |
| 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 128 |
| 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 256 |

## 22.4 Functional Description

This section provides a complete functional description of the timer TIM16B8CV2 block. Please refer to the detailed timer block diagram in Figure 22-30 as necessary.

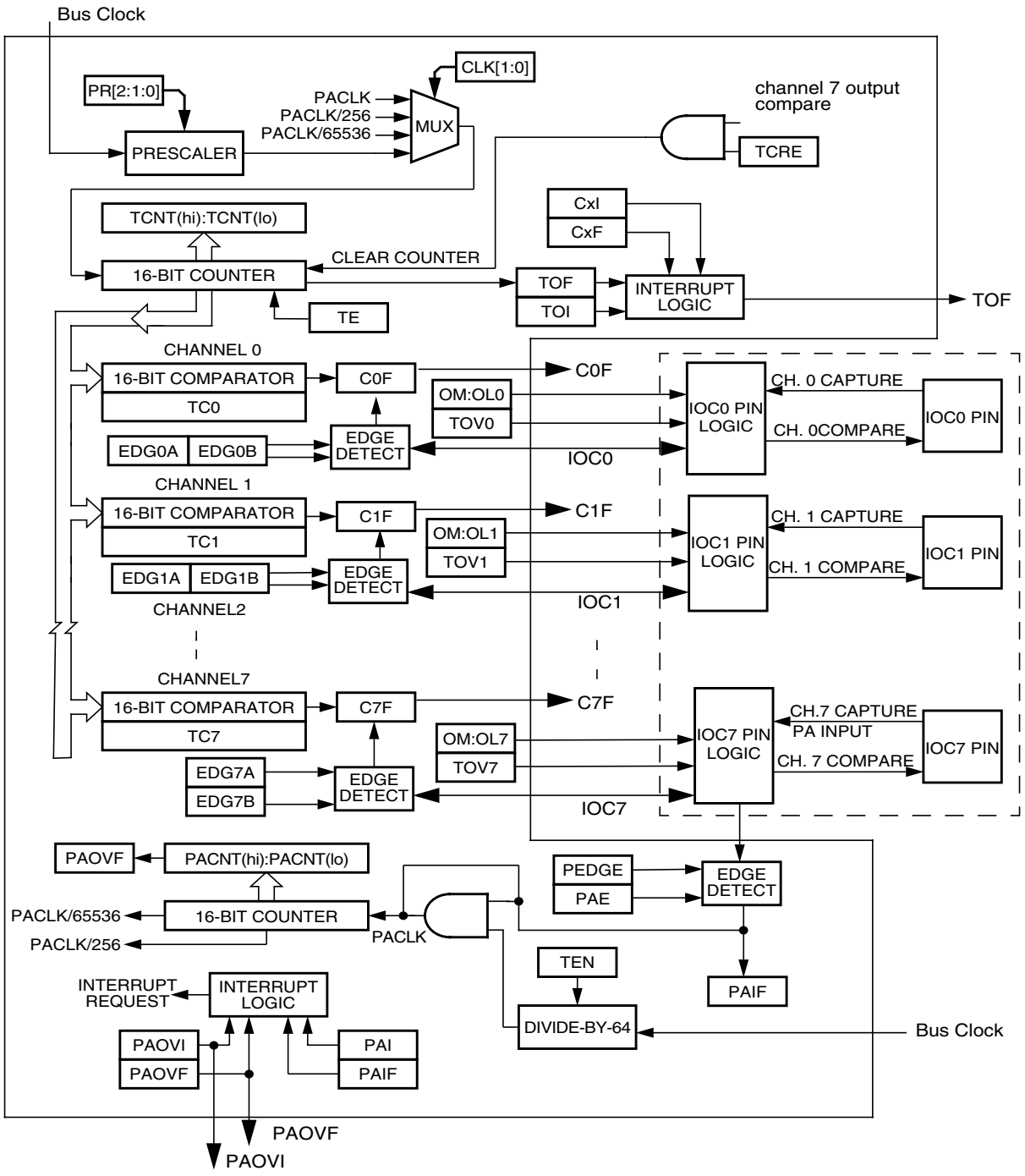



**Figure 22-30. Detailed Timer Block Diagram**

## 22.4.1 Prescaler

The prescaler divides the bus clock by 1,2,4,8,16,32,64 or 128. The prescaler select bits, PR[2:0], select the prescaler divisor. PR[2:0] are in timer system control register 2 (TSCR2).

The prescaler divides the bus clock by a prescalar value. Prescaler select bits PR[2:0] of in timer system control register 2 (TSCR2) are set to define a prescalar value that generates a divide by 1, 2, 4, 8, 16, 32, 64 and 128 when the PRNT bit in TSCR1 is disabled.

By enabling the PRNT bit of the TSCR1 register, the performance of the timer can be enhanced. In this case, it is possible to set additional prescaler settings for the main timer counter in the present timer by using PTPSR[7:0] bits of PTPSR register.

### 22.4.2 Input Capture

Clearing the I/O (input/output) select bit, IOSx, configures channel x as an input capture channel. The input capture function captures the time at which an external event occurs. When an active edge occurs on the pin of an input capture channel, the timer transfers the value in the timer counter into the timer channel registers, TCx.

The minimum pulse width for the input capture input is greater than two bus clocks.

An input capture on channel x sets the CxF flag. The CxI bit enables the CxF flag to generate interrupt requests. Timer module or Pulse Accumulator must stay enabled (TEN bit of TSCR1 or PAEN bit of PACTL regsiter must be set to one) while clearing CxF (writing one to CxF).

### 22.4.3 Output Compare

Setting the I/O select bit, IOSx, configures channel x as an output compare channel. The output compare function can generate a periodic pulse with a programmable polarity, duration, and frequency. When the timer counter reaches the value in the channel registers of an output compare channel, the timer can set, clear, or toggle the channel pin if the corresponding OCPDx bit is set to zero. An output compare on channel x sets the CxF flag. The CxI bit enables the CxF flag to generate interrupt requests. Timer module or Pulse Accumulator must stay enabled (TEN bit of TSCR1 or PAEN bit of PACTL regsiter must be set to one) while clearing CxF (writing one to CxF).

The output mode and level bits, OMx and OLx, select set, clear, toggle on output compare. Clearing both OMx and OLx results in no output compare action on the output compare channel pin.

Setting a force output compare bit, FOCx, causes an output compare on channel x. A forced output compare does not set the channel flag.

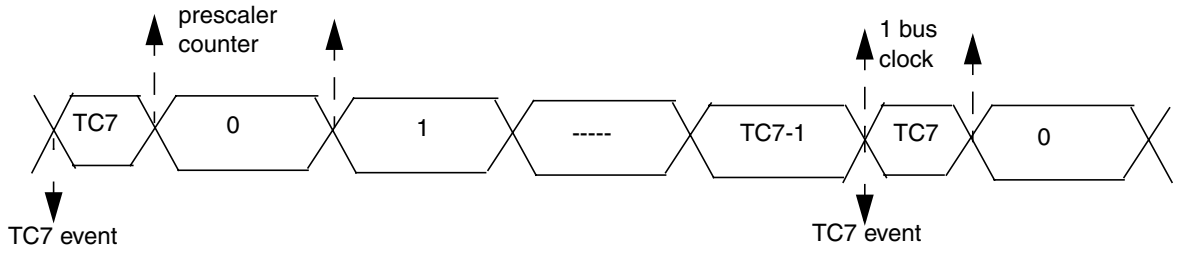
A channel 7 event, which can be a counter overflow when TTOV[7] is set or a successful output compare on channel 7, overrides output compares on all other output compare channels. The output compare 7 mask register masks the bits in the output compare 7 data register. The timer counter reset enable bit, TCRE, enables channel 7 output compares to reset the timer counter. A channel 7 output compare can reset the timer counter even if the IOC7 pin is being used as the pulse accumulator input.

Writing to the timer port bit of an output compare pin does not affect the pin state. The value written is stored in an internal latch. When the pin becomes available for general-purpose output, the last value written to the bit appears at the pin.

When TCRE is set and TC7 is not equal to 0, then TCNT will cycle from 0 to TC7. When TCNT reaches TC7 value, it lasts only one bus cycle then resets to 0.

Note: in Figure 22-31, if PR[2:0] is equal to 0, one prescaler counter is equal to one bus clock

**Figure 22-31. The TCNT cycle diagram under TCRE=1 condition**




### 22.4.3.1 OC Channel Initialization

Internal register whose output drives OCx can be programmed before timer drives OCx. The desired state can be programmed to this Internal register by writing a one to CFORCx bit with TIOSx, OCPDx and TEN bits set to one. Setting OCPDx to zero allows Interal register to drive the programmed state to OCx. This allows a glitch free switch over of port from general purpose I/O to timer output once the OCPDx bit is set to zero.

## 22.4.4 Pulse Accumulator

The pulse accumulator (PACNT) is a 16-bit counter that can operate in two modes:

Event counter mode — Counting edges of selected polarity on the pulse accumulator input pin, PAI.

Gated time accumulation mode — Counting pulses from a divide-by-64 clock. The PAMOD bit selects the mode of operation.

The minimum pulse width for the PAI input is greater than two bus clocks.

## 22.4.5 Event Counter Mode

Clearing the PAMOD bit configures the PACNT for event counter operation. An active edge on the IOC7 pin increments the pulse accumulator counter. The PEDGE bit selects falling edges or rising edges to increment the count.

**NOTE**

The PACNT input and timer channel 7 use the same pin IOC7. To use the IOC7, disconnect it from the output logic by clearing the channel 7 output mode and output level bits, OM7 and OL7. Also clear the channel 7 output compare 7 mask bit, OC7M7.

The Pulse Accumulator counter register reflect the number of active input edges on the PACNT input pin since the last reset.

The PAOVF bit is set when the accumulator rolls over from 0xFFFF to 0x0000. The pulse accumulator overflow interrupt enable bit, PAOVI, enables the PAOVF flag to generate interrupt requests.

**NOTE**

The pulse accumulator counter can operate in event counter mode even when the timer enable bit, TEN, is clear.

### 22.4.6 Gated Time Accumulation Mode

Setting the PAMOD bit configures the pulse accumulator for gated time accumulation operation. An active level on the PACNT input pin enables a divided-by-64 clock to drive the pulse accumulator. The PEDGE bit selects low levels or high levels to enable the divided-by-64 clock.

The trailing edge of the active level at the IOC7 pin sets the PAIF. The PAI bit enables the PAIF flag to generate interrupt requests.

The pulse accumulator counter register reflect the number of pulses from the divided-by-64 clock since the last reset.

**NOTE**

The timer prescaler generates the divided-by-64 clock. If the timer is not active, there is no divided-by-64 clock.

## 22.5 Resets

The reset state of each individual bit is listed within Section 22.3, “Memory Map and Register Definition” which details the registers and their bit fields.

## 22.6 Interrupts

This section describes interrupts originated by the TIM16B8CV2 block. Table 22-25 lists the interrupts generated by the TIM16B8CV2 to communicate with the MCU.

**Table 22-25. TIM16B8CV1 Interrupts**

| Interrupt | Offset (1) | Vector[1] | Priority[1] | Source | Description |
|---|---|---|---|---|---|
| C[7:0]F | — | — | — | Timer Channel 7–0 | Active high timer channel interrupts 7–0 |
| PAOVI | — | — | — | Pulse Accumulator Input | Active high pulse accumulator input interrupt |
| PAOVF | — | — | — | Pulse Accumulator Overflow | Pulse accumulator overflow interrupt |
| TOF | — | — | — | Timer Overflow | Timer Overflow interrupt |

1. Chip Dependent.

The TIM16B8CV2 uses a total of 11 interrupt vectors. The interrupt vector offsets and interrupt numbers are chip dependent.

### 22.6.1 Channel [7:0] Interrupt (C[7:0]F)

This active high outputs will be asserted by the module to request a timer channel 7 – 0 interrupt to be serviced by the system controller.

### 22.6.2 Pulse Accumulator Input Interrupt (PAOVI)

This active high output will be asserted by the module to request a timer pulse accumulator input interrupt to be serviced by the system controller.

### 22.6.3 Pulse Accumulator Overflow Interrupt (PAOVF)

This active high output will be asserted by the module to request a timer pulse accumulator overflow interrupt to be serviced by the system controller.

### 22.6.4 Timer Overflow Interrupt (TOF)

This active high output will be asserted by the module to request a timer overflow interrupt to be serviced by the system controller.

# Chapter 23
# Voltage Regulator (S12VREGL3V3V1)

**Table 23-1. Revision History**

| Revision Number | Revision Date | Sections Affected | Description of Changes |
|---|---|---|---|
| V01.02 | 09 Sep 2005 | 23.3.2.3/23-822 | - Updates for API external access and LVR flags. |
| V01.03 | 23 Sep 2005 | 23.3.2.1/23-820 | - VAE reset value is 1. |
| V01.04 | 08 Jun 2007 | 23.4.6/23-827 | - Added temperature sensor to customer information |

## 23.1 Introduction

Module VREG_3V3 is a tri output voltage regulator that provides two separate 1.84V (typical) supplies differing in the amount of current that can be sourced and a 2.82V (typical) supply. The regulator input voltage range is from 3.3V up to 5V (typical).

### 23.1.1 Features

Module VREG_3V3 includes these distinctive features:

- Three parallel, linear voltage regulators with bandgap reference
- Low-voltage detect (LVD) with low-voltage interrupt (LVI)
- Power-on reset (POR)
- Low-voltage reset (LVR)
- High Temperature Detect (HTD) with High Temperature Interrupt (HTI)
- Autonomous periodical interrupt (API)

### 23.1.2 Modes of Operation

There are three modes VREG_3V3 can operate in:

1. Full performance mode (FPM) (MCU is not in stop mode)

   The regulator is active, providing the nominal supply voltages with full current sourcing capability. Features LVD (low-voltage detect), LVR (low-voltage reset), and POR (power-on reset) and HTD (High Temperature Detect) are available. The API is available.
2. Reduced power mode (RPM) (MCU is in stop mode)

   The purpose is to reduce power consumption of the device. The output voltage may degrade to a lower value than in full performance mode, additionally the current sourcing capability is substantially reduced. Only the POR is available in this mode, LVD, LVR and HTD are disabled. The API is available.

3. Shutdown mode

   Controlled by VREGEN (see device level specification for connectivity of VREGEN).

   This mode is characterized by minimum power consumption. The regulator outputs are in a high-impedance state, only the POR feature is available, LVD, LVR and HTD are disabled. The API internal RC oscillator clock is not available.

   This mode must be used to disable the chip internal regulator VREG_3V3, i.e., to bypass the VREG_3V3 to use external supplies.

### 23.1.3 Block Diagram

Figure 23-1 shows the function principle of VREG_3V3 by means of a block diagram. The regulator core REG consists of three parallel subblocks, REG1, REG2 and REG3, providing three independent output voltages.

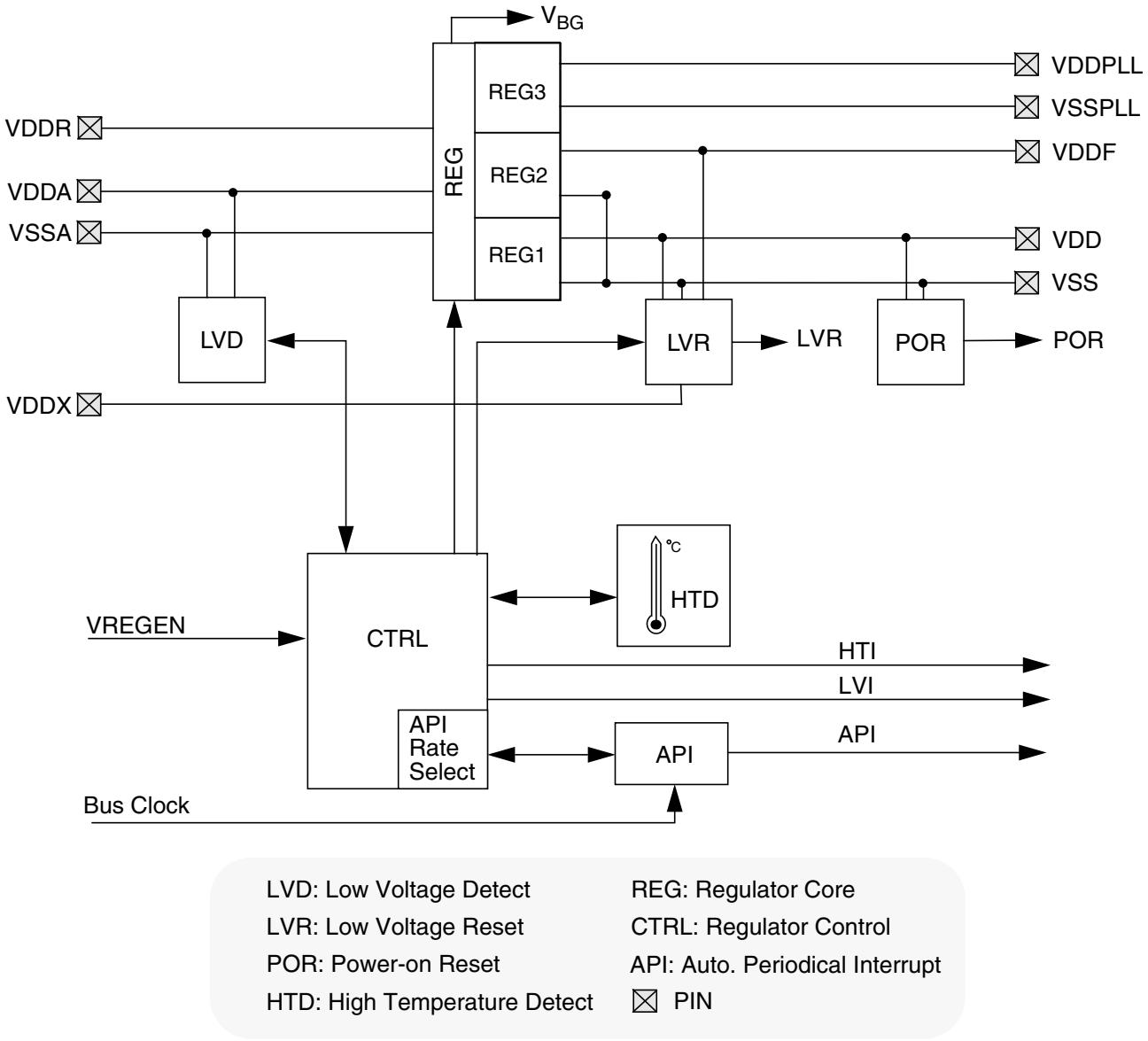



**Figure 23-1. VREG_3V3 Block Diagram**

## 23.2 External Signal Description

Due to the nature of VREG_3V3 being a voltage regulator providing the chip internal power supply voltages, most signals are power supply signals connected to pads.

Table 23-2 shows all signals of VREG_3V3 associated with pins.

**Table 23-2. Signal Properties**

| Name | Function | Reset State | Pull Up |
|---|---|---|---|
| VDDR | Power input (positive supply) | — | — |
| VDDA | Quiet input (positive supply) | — | — |
| VSSA | Quiet input (ground) | — | — |
| VDDX | Power input (positive supply) | — | — |
| VDD | Primary output (positive supply) | — | — |
| VSS | Primary output (ground) | — | — |
| VDDF | Secondary output (positive supply) | — | — |
| VDDPLL | Tertiary output (positive supply) | — | — |
| VSSPLL | Tertiary output (ground) | — | — |
| VREGEN (optional) | Optional Regulator Enable | — | — |
| VREG_API (optional) | VREG Autonomous Periodical Interrupt output | — | — |

**NOTE**

Check device level specification for connectivity of the signals.

### 23.2.1 VDDR — Regulator Power Input Pins

Signal VDDR is the power input of VREG_3V3. All currents sourced into the regulator loads flow through this pin. A chip external decoupling capacitor (100 nF...220 nF, X7R ceramic) between VDDR and VSSR (if VSSR is not available VSS) can smooth ripple on VDDR.

For entering Shutdown Mode, pin VDDR should also be tied to ground on devices without VREGEN pin.

### 23.2.2 VDDA, VSSA — Regulator Reference Supply Pins

Signals VDDA/VSSA, which are supposed to be relatively quiet, are used to supply the analog parts of the regulator. Internal precision reference circuits are supplied from these signals. A chip external decoupling capacitor (100 nF...220 nF, X7R ceramic) between VDDA and VSSA can further improve the quality of this supply.

### 23.2.3 VDD, VSS — Regulator Output1 (Core Logic) Pins

Signals VDD/VSS are the primary outputs of VREG_3V3 that provide the power supply for the core logic. These signals are connected to device pins to allow external decoupling capacitors (220 nF, X7R ceramic).

In Shutdown Mode an external supply driving VDD/VSS can replace the voltage regulator.

### 23.2.4 VDDF — Regulator Output2 (NVM Logic) Pins

Signals VDDF/VSS are the secondary outputs of VREG_3V3 that provide the power supply for the NVM logic. These signals are connected to device pins to allow external decoupling capacitors (220 nF, X7R ceramic).

In Shutdown Mode an external supply driving VDDF/VSS can replace the voltage regulator.

### 23.2.5 VDDPLL, VSSPLL — Regulator Output3 (PLL) Pins

Signals VDDPLL/VSSPLL are the secondary outputs of VREG_3V3 that provide the power supply for the PLL and oscillator. These signals are connected to device pins to allow external decoupling capacitors (100 nF...220 nF, X7R ceramic).

In Shutdown Mode, an external supply driving VDDPLL/VSSPLL can replace the voltage regulator.

### 23.2.6 VDDX — Power Input Pin

Signals VDDX/VSS are monitored by VREG_3V3 with the LVR feature.

### 23.2.7 VREGEN — Optional Regulator Enable Pin

This optional signal is used to shutdown VREG_3V3. In that case, VDD/VSS and VDDPLL/VSSPLL must be provided externally. Shutdown mode is entered with VREGEN being low. If VREGEN is high, the VREG_3V3 is either in Full Performance Mode or in Reduced Power Mode.

For the connectivity of VREGEN, see device specification.

**NOTE**

Switching from FPM or RPM to shutdown of VREG_3V3 and vice versa is not supported while MCU is powered.

### 23.2.8 VREG_API — Optional Autonomous Periodical Interrupt Output Pin

This pin provides the signal selected via APIEA if system is set accordingly. See 23.3.2.3, “Autonomous Periodical Interrupt Control Register (VREGAPICL) and 23.4.8, “Autonomous Periodical Interrupt (API) for details.

For the connectivity of VREG_API, see device specification.

## 23.3 Memory Map and Register Definition

This section provides a detailed description of all registers accessible in VREG_3V3.

If enabled in the system, the VREG_3V3 will abort all read and write accesses to reserved registers within it’s memory slice. See device level specification for details.

## 23.3.1 Module Memory Map

A summary of the registers associated with the VREG_3V3 sub-block is shown in Figure 23-2. Detailed descriptions of the registers and bits are given in the subsections that follow

**Figure 23-2. Register Summary**

| Address | Name | | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|---|---|
| 0x02F0 | VREGHTCL | R | 0 | 0 | VSEL | VAE | HTEN | HTDS | HTIE | HTIF |
| | | W | | | | | | | | |
| 0x02F1 | VREGCTRL | R | 0 | 0 | 0 | 0 | 0 | LVDS | LVIE | LVIF |
| | | W | | | | | | | | |
| 0x02F2 | VREGAPICL | R | APICLK | 0 | 0 | APIFES | APIEA | APIFE | APIE | APIF |
| | | W | | | | | | | | |
| 0x02F3 | VREGAPITR | R | APITR5 | APITR4 | APITR3 | APITR2 | APITR1 | APITR0 | 0 | 0 |
| | | W | | | | | | | | |
| 0x02F4 | VREGAPIRH | R | APIR15 | APIR14 | APIR13 | APIR12 | APIR11 | APIR10 | APIR9 | APIR8 |
| | | W | | | | | | | | |
| 0x02F5 | VREGAPIRL | R | APIR7 | APIR6 | APIR5 | APIR4 | APIR3 | APIR2 | APIR1 | APIR0 |
| | | W | | | | | | | | |
| 0x02F6 | Reserved 06 | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | W | | | | | | | | |
| 0x02F7 | VREGHTTR | R | HTOEN | 0 | 0 | 0 | HTTR3 | HTTR2 | HTTR1 | HTTR0 |
| | | W | | | | | | | | |

## 23.3.2 Register Descriptions

This section describes all the VREG_3V3 registers and their individual bits.

### 23.3.2.1 High Temperature Control Register (VREGHTCL)

The VREGHTCL register allows to configure the VREG temperature sense features.

0x02F0

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | VSEL | VAE | HTEN | HTDS | HTIE | HTIF |
| W | | | | | | | | |
| Reset | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |

= Unimplemented or Reserved

**Table 23-3. VREGHTCL Field Descriptions**

| Field | Description |
|---|---|
| 7, 6<br>Reserved | These reserved bits are used for test purposes and writable only in special modes.<br>They must remain clear for correct temperature sensor operation. |
| 5<br>VSEL | **Voltage Access Select Bit** — If set, the bandgap reference voltage $V_{BG}$ can be accessed internally (i.e. multiplexed to an internal Analog to Digital Converter channel). The internal access must be enabled by bit VAE. See device level specification for connectivity.<br>0 An internal temperature proportional voltage $V_{HT}$ can be accessed internally if VAE is set.<br>1 Bandgap reference voltage $V_{BG}$ can be accessed internally if VAE is set. |
| 4<br>VAE | **Voltage Access Enable Bit** — If set, the voltage selected by bit VSEL can be accessed internally (i.e. multiplexed to an internal Analog to Digital Converter channel). See device level specification for connectivity.<br>0 Voltage selected by VSEL can not be accessed internally (i.e. External analog input is connected to Analog to Digital Converter channel).<br>1 Voltage selected by VSEL can be accessed internally. |
| 3<br>HTEN | **High Temperature Enable Bit** — If set the temperature sense is enabled.<br>0 The temperature sense is disabled.<br>1 The temperature sense is enabled. |
| 2<br>HTDS | **High Temperature Detect Status Bit** —<br>This read-only status bit reflects the temperature status. Writes have no effect.<br>0 Temperature $T_{DIE}$ is below level $T_{HTID}$ or RPM or Shutdown Mode.<br>1 Temperature $T_{DIE}$ is above level $T_{HTIA}$ and FPM. |
| 1<br>HTIE | **High Temperature Interrupt Enable Bit**<br>0 Interrupt request is disabled.<br>1 Interrupt will be requested whenever HTIF is set. |
| 0<br>HTIF | **High Temperature Interrupt Flag** — HTIF — High Temperature Interrupt Flag<br>HTIF is set to 1 when HTDS status bit changes. This flag can only be cleared by writing a 1. Writing a 0 has no effect. If enabled (HTIE=1), HTIF causes an interrupt request.<br>0 No change in HTDS bit.<br>1 HTDS bit has changed.<br>**Note:** On entering the reduced power mode the HTIF is not cleared by the VREG. |

## 23.3.2.2 Control Register (VREGCTRL)

The VREGCTRL register allows the configuration of the VREG_3V3 low-voltage detect features.

0x02F1

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | 0 | 0 | 0 | LVDS | LVIE | LVIF |
| W | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented or Reserved

**Figure 23-3. Control Register (VREGCTRL)**

**Table 23-4. VREGCTRL Field Descriptions**

| Field | Description |
|---|---|
| 2<br>LVDS | **Low-Voltage Detect Status Bit** — This read-only status bit reflects the input voltage. Writes have no effect.<br>0 Input voltage $V_{DDA}$ is above level $V_{LVID}$ or RPM or shutdown mode.<br>1 Input voltage $V_{DDA}$ is below level $V_{LVIA}$ and FPM. |
| 1<br>LVIE | **Low-Voltage Interrupt Enable Bit**<br>0 Interrupt request is disabled.<br>1 Interrupt will be requested whenever LVIF is set. |
| 0<br>LVIF | **Low-Voltage Interrupt Flag** — LVIF is set to 1 when LVDS status bit changes. This flag can only be cleared by writing a 1. Writing a 0 has no effect. If enabled (LVIE = 1), LVIF causes an interrupt request.<br>0 No change in LVDS bit.<br>1 LVDS bit has changed.<br>**Note:** On entering the Reduced Power Mode the LVIF is not cleared by the VREG_3V3. |

### 23.3.2.3 Autonomous Periodical Interrupt Control Register (VREGAPICL)

The VREGAPICL register allows the configuration of the VREG_3V3 autonomous periodical interrupt features.

0x02F2

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | APICLK | 0 | 0 | APIES | APIEA | APIFE | APIE | APIF |
| W | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented or Reserved

**Figure 23-4. Autonomous Periodical Interrupt Control Register (VREGAPICL)**

**Table 23-5. VREGAPICL Field Descriptions**

| Field | Description |
|---|---|
| 7<br>APICLK | **Autonomous Periodical Interrupt Clock Select Bit** — Selects the clock source for the API. Writable only if APIFE = 0; APICLK cannot be changed if APIFE is set by the same write operation.<br>0 Autonomous periodical interrupt clock used as source.<br>1 Bus clock used as source. |
| 4<br>APIES | **Autonomous Periodical Interrupt External Select Bit** — Selects the waveform at the external pin.If set, at the external pin a clock is visible with 2 times the selected API Period (Table 23-9). If not set, at the external pin will be a high pulse at the end of every selected period with the size of half of the min period (Table 23-9). See device level specification for connectivity.<br>0 At the external periodic high pulses are visible, if APIEA and APIFE is set.<br>1 At the external pin a clock is visible, if APIEA and APIFE is set. |
| 3<br>APIEA | **Autonomous Periodical Interrupt External Access Enable Bit** — If set, the waveform selected by bit APIES can be accessed externally. See device level specification for connectivity.<br>0 Waveform selected by APIES can not be accessed externally.<br>1 Waveform selected by APIES can be accessed externally, if APIFE is set. |
| 2<br>APIFE | **Autonomous Periodical Interrupt Feature Enable Bit** — Enables the API feature and starts the API timer when set.<br>0 Autonomous periodical interrupt is disabled.<br>1 Autonomous periodical interrupt is enabled and timer starts running. |

**Table 23-5. VREGAPICL Field Descriptions (continued)**

| Field | Description |
|---|---|
| 1<br>APIE | **Autonomous Periodical Interrupt Enable Bit**<br>0 API interrupt request is disabled.<br>1 API interrupt will be requested whenever APIF is set. |
| 0<br>APIF | **Autonomous Periodical Interrupt Flag** — APIF is set to 1 when the in the API configured time has elapsed. This flag can only be cleared by writing a 1 to it. Clearing of the flag has precedence over setting. Writing a 0 has no effect. If enabled (APIE = 1), APIF causes an interrupt request.<br>0 API timeout has not yet occurred.<br>1 API timeout has occurred. |

### 23.3.2.4 Autonomous Periodical Interrupt Trimming Register (VREGAPITR)

The VREGAPITR register allows to trim the API timeout period.

0x02F3

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | APITR5 | APITR4 | APITR3 | APITR2 | APITR1 | APITR0 | 0 | 0 |
| W | | | | | | | | |
| Reset | 0[1] | 0[1] | 0[1] | 0[1] | 0[1] | 0[1] | 0 | 0 |

1. Reset value is either 0 or preset by factory. See Section 1 (Device Overview) for details.

= Unimplemented or Reserved

**Figure 23-5. Autonomous Periodical Interrupt Trimming Register (VREGAPITR)**

**Table 23-6. VREGAPITR Field Descriptions**

| Field | Description |
|---|---|
| 7–2<br>APITR[5:0] | **Autonomous Periodical Interrupt Period Trimming Bits** — See Table 23-7 for trimming effects. |

**Table 23-7. Trimming Effect of APIT**

| Bit | Trimming Effect |
|---|---|
| APITR[5] | Increases period |
| APITR[4] | Decreases period less than APITR[5] increased it |
| APITR[3] | Decreases period less than APITR[4] |
| APITR[2] | Decreases period less than APITR[3] |
| APITR[1] | Decreases period less than APITR[2] |
| APITR[0] | Decreases period less than APITR[1] |

### 23.3.2.5 Autonomous Periodical Interrupt Rate High and Low Register (VREGAPIRH / VREGAPIRL)

The VREGAPIRH and VREGAPIRL register allows the configuration of the VREG_3V3 autonomous periodical interrupt rate.

0x02F4

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | APIR15 | APIR14 | APIR13 | APIR12 | APIR11 | APIR10 | APIR9 | APIR8 |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented or Reserved

**Figure 23-6. Autonomous Periodical Interrupt Rate High Register (VREGAPIRH)**

0x02F5

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | APIR7 | APIR6 | APIR5 | APIR4 | APIR3 | APIR2 | APIR1 | APIR0 |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 23-7. Autonomous Periodical Interrupt Rate Low Register (VREGAPIRL)**

**Table 23-8. VREGAPIRH / VREGAPIRL Field Descriptions**

| Field | Description |
|---|---|
| 15-0<br>APIR[15:0] | **Autonomous Periodical Interrupt Rate Bits** — These bits define the timeout period of the API. See Table 23-9 for details of the effect of the autonomous periodical interrupt rate bits. Writable only if APIFE = 0 of VREGAPICL register. |

**Table 23-9. Selectable Autonomous Periodical Interrupt Periods**

| APICLK | APIR[15:0] | Selected Period |
|---|---|---|
| 0 | 0000 | 0.2 ms(1) |
| 0 | 0001 | 0.4 ms[1] |
| 0 | 0002 | 0.6 ms[1] |
| 0 | 0003 | 0.8 ms[1] |
| 0 | 0004 | 1.0 ms[1] |
| 0 | 0005 | 1.2 ms[1] |
| 0 | ..... | ..... |
| 0 | FFFD | 13106.8 ms[1] |
| 0 | FFFE | 13107.0 ms[1] |
| 0 | FFFF | 13107.2 ms[1] |
| 1 | 0000 | 2 * bus clock period |
| 1 | 0001 | 4 * bus clock period |
| 1 | 0002 | 6 * bus clock period |
| 1 | 0003 | 8 * bus clock period |
| 1 | 0004 | 10 * bus clock period |
| 1 | 0005 | 12 * bus clock period |
| 1 | ..... | ..... |
| 1 | FFFD | 131068 * bus clock period |
| 1 | FFFE | 131070 * bus clock period |

**Table 23-9. Selectable Autonomous Periodical Interrupt Periods (continued)**

| APICLK | APIR[15:0] | Selected Period |
|---|---|---|
| 1 | FFFF | 131072 * bus clock period |

1. When trimmed within specified accuracy. See electrical specifications for details.

The period can be calculated as follows depending of APICLK:

Period = 2*(APIR[15:0] + 1) * 0.1 ms or period = 2*(APIR[15:0] + 1) * bus clock period

### 23.3.2.6 Reserved 06

The Reserved 06 is reserved for test purposes.

0x02F6

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| W | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented or Reserved

**Figure 23-8. Reserved 06**

### 23.3.2.7 High Temperature Trimming Register (VREGHTTR)

The VREGHTTR register allows to trim the VREG temperature sense.

0x02F7

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | HTOEN | 0 | 0 | 0 | HTTR3 | HTTR2 | HTTR1 | HTTR0 |
| W | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0[1] | 0[1] | 0[1] | 0[1] |

1. Reset value is either 0 or preset by factory. See Section 1 (Device Overview) for details.

= Unimplemented or Reserved

**Figure 23-9. VREGHTTR**

**Table 23-10. VREGHTTR field descriptions**

| Field | Description |
|---|---|
| 7<br>HTOEN | **High Temperature Offset Enable Bit** — If set the temperature sense offset is enabled<br>0 The temperature sense offset is disabled<br>1 The temperature sense offset is enabled |
| 3–0<br>HTTR[3:0] | **High Temperature Trimming Bits** — See Table 23-11 for trimming effects. |

**Table 23-11. Trimming Effect**

| Bit | Trimming Effect |
|---|---|
| HTTR[3] | Increases $V_{HT}$ twice of HTTR[2] |

**Table 23-11. Trimming Effect (continued)**

| Bit | Trimming Effect |
|---|---|
| HTTR[2] | Increases $V_{HT}$ twice of HTTR[1] |
| HTTR[1] | Increases $V_{HT}$ twice of HTTR[0] |
| HTTR[0] | Increases $V_{HT}$ (to compensate Temperature Offset) |

# 23.4 Functional Description

## 23.4.1 General

Module VREG_3V3 is a voltage regulator, as depicted in Figure 23-1. The regulator functional elements are the regulator core (REG), a low-voltage detect module (LVD), a control block (CTRL), a power-on reset module (POR), and a low-voltage reset module (LVR)and a high temperature sensor (HTD).

## 23.4.2 Regulator Core (REG)

Respectively its regulator core has three parallel, independent regulation loops (REG1,REG2 and REG3). REG1 and REG3 differ only in the amount of current that can be delivered.

The regulators are linear regulator with a bandgap reference when operated in Full Performance Mode. They act as a voltage clamp in Reduced Power Mode. All load currents flow from input VDDR to VSS or VSSPLL. The reference circuits are supplied by VDDA and VSSA.

### 23.4.2.1 Full Performance Mode

In Full Performance Mode, the output voltage is compared with a reference voltage by an operational amplifier. The amplified input voltage difference drives the gate of an output transistor.

### 23.4.2.2 Reduced Power Mode

In Reduced Power Mode, the gate of the output transistor is connected directly to a reference voltage to reduce power consumption. Mode switching from reduced power to full performance requires a transition time of $t_{vup}$, if the voltage regulator is enabled.

## 23.4.3 Low-Voltage Detect (LVD)

Subblock LVD is responsible for generating the low-voltage interrupt (LVI). LVD monitors the input voltage ($V_{DDA}$–$V_{SSA}$) and continuously updates the status flag LVDS. Interrupt flag LVIF is set whenever status flag LVDS changes its value. The LVD is available in FPM and is inactive in Reduced Power Mode or Shutdown Mode.

### 23.4.4 Power-On Reset (POR)

This functional block monitors VDD. If $V_{DD}$ is below $V_{PORD}$, POR is asserted; if $V_{DD}$ exceeds $V_{PORD}$, the POR is deasserted. POR asserted forces the MCU into Reset. POR Deasserted will trigger the power-on sequence.

### 23.4.5 Low-Voltage Reset (LVR)

Block LVR monitors the supplies VDD, VDDX and VDDF. If one (or more) drops below it's corresponding assertion level, signal LVR asserts; if all VDD,VDDX and VDDF supplies are above their corresponding deassertion levels, signal LVR deasserts. The LVR function is available only in Full Performance Mode.

### 23.4.6 HTD - High Temperature Detect

Subblock HTD is responsible for generating the high temperature interrupt (HTI). HTD monitors the die temperature $T_{DIE}$ and continuously updates the status flag HTDS.
Interrupt flag HTIF is set whenever status flag HTDS changes its value.

The HTD is available in FPM and is inactive in Reduced Power Mode and Shutdown Mode.

The HT Trimming bits HTTR[3:0] can be set so that the temperature offset is zero, if accurate temperature measurement is desired.

See Table 23-11 for the trimming effect of APITR.

### 23.4.7 Regulator Control (CTRL)

This part contains the register block of VREG_3V3 and further digital functionality needed to control the operating modes. CTRL also represents the interface to the digital core logic.

### 23.4.8 Autonomous Periodical Interrupt (API)

Subblock API can generate periodical interrupts independent of the clock source of the MCU. To enable the timer, the bit APIFE needs to be set.

The API timer is either clocked by a trimmable internal RC oscillator or the bus clock. Timer operation will freeze when MCU clock source is selected and bus clock is turned off. See CRG specification for details. The clock source can be selected with bit APICLK. APICLK can only be written when APIFE is not set.

The APIR[15:0] bits determine the interrupt period. APIR[15:0] can only be written when APIFE is cleared. As soon as APIFE is set, the timer starts running for the period selected by APIR[15:0] bits. When the configured time has elapsed, the flag APIF is set. An interrupt, indicated by flag APIF = 1, is triggered if interrupt enable bit APIE = 1. The timer is started automatically again after it has set APIF.

The procedure to change APICLK or APIR[15:0] is first to clear APIFE, then write to APICLK or APIR[15:0], and afterwards set APIFE.

The API Trimming bits APITR[5:0] must be set so the minimum period equals 0.2 ms if stable frequency is desired.

See Table 23-7 for the trimming effect of APITR.

**NOTE**

The first period after enabling the counter by APIFE might be reduced by API start up delay $t_{sdel}$. The API internal RC oscillator clock is not available if VREG_3V3 is in Shutdown Mode.

It is possible to generate with the API a waveform at an external pin by enabling the API by setting APIFE and enabling the external access with setting APIEA. By setting APIES the waveform can be selected. If APIES is set, then at the external pin a clock is visible with 2 times the selected API Period (Table 23-9). If APIES is not set, then at the external pin will be a high pulse at the end of every selected period with the size of half of the min period (Table 23-9). See device level specification for connectivity.

## 23.4.9 Resets

This section describes how VREG_3V3 controls the reset of the MCU.The reset values of registers and signals are provided in Section 23.3, “Memory Map and Register Definition”. Possible reset sources are listed in Table 23-12.

**Table 23-12. Reset Sources**

| Reset Source | Local Enable |
|---|---|
| Power-on reset | Always active |
| Low-voltage reset | Available only in Full Performance Mode |

## 23.4.10 Description of Reset Operation

### 23.4.10.1 Power-On Reset (POR)

During chip power-up the digital core may not work if its supply voltage $V_{DD}$ is below the POR deassertion level ($V_{PORD}$). Therefore, signal POR, which forces the other blocks of the device into reset, is kept high until $V_{DD}$ exceeds $V_{PORD}$. The MCU will run the start-up sequence after POR deassertion. The power-on reset is active in all operation modes of VREG_3V3.

### 23.4.10.2 Low-Voltage Reset (LVR)

For details on low-voltage reset, see Section 23.4.5, “Low-Voltage Reset (LVR)”.

## 23.4.11 Interrupts

This section describes all interrupts originated by VREG_3V3.

The interrupt vectors requested by VREG_3V3 are listed in Table 23-13. Vector addresses and interrupt priorities are defined at MCU level.

**Table 23-13. Interrupt Vectors**

| Interrupt Source | Local Enable |
|---|---|
| Low-voltage interrupt (LVI) | LVIE = 1; available only in Full Performance Mode |
| High Temperature Interrupt (HTI) | HTIE=1; available only in Full Performance Mode |
| Autonomous periodical interrupt (API) | APIE = 1 |

### 23.4.11.1 Low-Voltage Interrupt (LVI)

In FPM, VREG_3V3 monitors the input voltage $V_{DDA}$. Whenever $V_{DDA}$ drops below level $V_{LVIA,}$ the status bit LVDS is set to 1. On the other hand, LVDS is reset to 0 when $V_{DDA}$ rises above level $V_{LVID}$. An interrupt, indicated by flag LVIF = 1, is triggered by any change of the status bit LVDS if interrupt enable bit LVIE = 1.

**NOTE**

On entering the Reduced Power Mode, the LVIF is not cleared by the VREG_3V3.

### 23.4.11.2 HTI - High Temperature Interrupt

In FPM VREG monitors the die temperature $T_{DIE}$. Whenever $T_{DIE}$ exceeds level $T_{HTIA}$ the status bit HTDS is set to 1. Vice versa, HTDS is reset to 0 when $T_{DIE}$ get below level $T_{HTID}$. An interrupt, indicated by flag HTIF=1, is triggered by any change of the status bit HTDS if interrupt enable bit HTIE=1.

**NOTE**

On entering the Reduced Power Mode the HTIF is not cleared by the VREG.

### 23.4.11.3 Autonomous Periodical Interrupt (API)

As soon as the configured timeout period of the API has elapsed, the APIF bit is set. An interrupt, indicated by flag APIF = 1, is triggered if interrupt enable bit APIE = 1.

# Chapter 24
# 128 KByte Flash Module (S12XFTM128K2V1)

**Table 24-1. Revision History**

| Revision Number | Revision Date | Sections Affected | Description of Changes |
|---|---|---|---|
| V01.10 | 30 Nov 2007 | 24.1.3/24-834 | - Correction toTable 24-6 |
| V01.11 | 19 Dec 2007 | 24.4.2/24-867<br>24.4.2/24-867<br>24.4.2/24-867<br>24.4.2/24-867<br>24.3.1/24-836<br>24.1.3/24-834<br>24.3.1/24-836<br>24.1.2.2/24-833<br>24.3.1/24-836 | - Removed Load Data Field command 0x05<br>- Updated Command Error Handling tables based on parent-child relationship with FTM256K2<br>- Corrected Error Handling table for Full Partition D-Flash, Partition D-Flash, and EEPROM Emulation Query commands<br>- Corrected maximum allowed ERPART for Full Partition D-Flash and Partition D-Flash commands<br>- Corrected P-Flash IFR Accessibility table<br>- Corrected Buffer RAM size in Feature List<br>- Corrected EEE Resource Memory Map<br>- Changed D-Flash size from 16Kbytes to 32Kbytes<br>- Corrected P-Flash Memory Map |
| V01.12 | 25 Sep 2009 | 24.1/24-832<br>24.3.2.1/24-843<br>24.4.2.4/24-870<br>24.4.2.6/24-871<br>24.4.2.11/24-875<br>24.4.2.11/24-875<br>24.4.2.11/24-875<br>24.4.2.19/24-884<br>24.3.2/24-841<br>24.3.2.1/24-843<br>24.4.1.2/24-862<br>24.6/24-890 | - Change references for D-Flash from 16 Kbytes to 32 Kbytes<br>- Clarify single bit fault correction for P-Flash phrase<br>- Expand FDIV vs OSCCLK Frequency table<br>- Add statement concerning code runaway when executing Read Once command from Flash block containing associated fields<br>- Add statement concerning code runaway when executing Program Once command from Flash block containing associated fields<br>- Add statement concerning code runaway when executing Verify Backdoor Access Key command from Flash block containing associated fields<br>- Relate Key 0 to associated Backdoor Comparison Key address<br>- Change “power down reset” to “reset”<br>- Add ACCERR condition for Disable EEPROM Emulation command<br>The following changes were made to clarify module behavior related to Flash register access during reset sequence and while Flash commands are active:<br>- Add caution concerning register writes while command is active<br>- Writes to FCLKDIV are allowed during reset sequence while CCIF is clear<br>- Add caution concerning register writes while command is active<br>- Writes to FCCOBIX, FCCOBHI, FCCOBLO registers are ignored during reset sequence |

## 24.1 Introduction

The FTM128K2 module implements the following:

- 128 Kbytes of P-Flash (Program Flash) memory, consisting of 2 physical Flash blocks, intended primarily for nonvolatile code storage
- 32 Kbytes of D-Flash (Data Flash) memory, consisting of 1 physical Flash block, that can be used as nonvolatile storage to support the built-in hardware scheme for emulated EEPROM, as basic Flash memory primarily intended for nonvolatile data storage, or as a combination of both
- 2 Kbytes of buffer RAM, consisting of 1 physical RAM block, that can be used as emulated EEPROM using a built-in hardware scheme, as basic RAM, or as a combination of both

The Flash memory is ideal for single-supply applications allowing for field reprogramming without requiring external high voltage sources for program or erase operations. The Flash module includes a memory controller that executes commands to modify Flash memory contents or configure module resources for emulated EEPROM operation. The user interface to the memory controller consists of the indexed Flash Common Command Object (FCCOB) register which is written to with the command, global address, data, and any required command parameters. The memory controller must complete the execution of a command before the FCCOB register can be written to with a new command.

**CAUTION**

A Flash word or phrase must be in the erased state before being programmed. Cumulative programming of bits within a Flash word or phrase is not allowed.

The RAM and Flash memory may be read as bytes, aligned words, or misaligned words. Read access time is one bus cycle for bytes and aligned words, and two bus cycles for misaligned words. For Flash memory, an erased bit reads 1 and a programmed bit reads 0.

It is not possible to read from a Flash block while any command is executing on that specific Flash block. It is possible to read from a Flash block while a command is executing on a different Flash block.

Both P-Flash and D-Flash memories are implemented with Error Correction Codes (ECC) that can resolve single bit faults and detect double bit faults. For P-Flash memory, the ECC implementation requires that programming be done on an aligned 8 byte basis (a Flash phrase). Since P-Flash memory is always read by phrase, only one single bit fault in the phrase containing the byte or word accessed will be corrected.

### 24.1.1 Glossary

**Buffer RAM** — The buffer RAM constitutes the volatile memory store required for EEE. Memory space in the buffer RAM not required for EEE can be partitioned to provide volatile memory space for applications.

**Command Write Sequence** — An MCU instruction sequence to execute built-in algorithms (including program and erase) on the Flash memory.

**D-Flash Memory** — The D-Flash memory constitutes the nonvolatile memory store required for EEE. Memory space in the D-Flash memory not required for EEE can be partitioned to provide nonvolatile memory space for applications.

**D-Flash Sector** — The D-Flash sector is the smallest portion of the D-Flash memory that can be erased. The D-Flash sector consists of four 64 byte rows for a total of 256 bytes.

**EEE (Emulated EEPROM)** — A method to emulate the small sector size features and endurance characteristics associated with an EEPROM.

**EEE IFR** — Nonvolatile information register located in the D-Flash block that contains data required to partition the D-Flash memory and buffer RAM for EEE. The EEE IFR is visible in the global memory map by setting the EEEIFRON bit in the MMCCTL1 register.

**NVM Command Mode** — An NVM mode using the CPU to setup the FCCOB register to pass parameters required for Flash command execution.

**Phrase** — An aligned group of four 16-bit words within the P-Flash memory. Each phrase includes eight ECC bits for single bit fault correction and double bit fault detection within the phrase.

**P-Flash Memory** — The P-Flash memory constitutes the main nonvolatile memory store for applications.

**P-Flash Sector** — The P-Flash sector is the smallest portion of the P-Flash memory that can be erased. Each P-Flash sector contains 1024 bytes.

**Program IFR** — Nonvolatile information register located in the P-Flash block that contains the Device ID, Version ID, and the Program Once field. The Program IFR is visible in the global memory map by setting the PGMIFRON bit in the MMCCTL1 register.

## 24.1.2 Features

### 24.1.2.1 P-Flash Features

- 128 Kbytes of P-Flash memory composed of two 64 Kbyte Flash blocks. The 64 Kbyte Flash blocks are each divided into 64 sectors of 1024 bytes.
- Single bit fault correction and double bit fault detection within a 64-bit phrase during read operations
- Automated program and erase algorithm with verify and generation of ECC parity bits
- Fast sector erase and phrase program operation
- Ability to program up to one phrase in each P-Flash block simultaneously
- Flexible protection scheme to prevent accidental program or erase of P-Flash memory

### 24.1.2.2 D-Flash Features

- Up to 32 Kbytes of D-Flash memory with 256 byte sectors for user access
- Dedicated commands to control access to the D-Flash memory over EEE operation

- Single bit fault correction and double bit fault detection within a word during read operations
- Automated program and erase algorithm with verify and generation of ECC parity bits
- Fast sector erase and word program operation
- Ability to program up to four words in a burst sequence

#### 24.1.2.3 Emulated EEPROM Features

- Up to 2 Kbytes of emulated EEPROM (EEE) accessible as 2 Kbytes of RAM
- Flexible protection scheme to prevent accidental program or erase of data
- Automatic EEE file handling using an internal Memory Controller
- Automatic transfer of valid EEE data from D-Flash memory to buffer RAM on reset
- Ability to monitor the number of outstanding EEE related buffer RAM words left to be programmed into D-Flash memory
- Ability to disable EEE operation and allow priority access to the D-Flash memory
- Ability to cancel all pending EEE operations and allow priority access to the D-Flash memory

#### 24.1.2.4 User Buffer RAM Features

- Up to 2 Kbytes of RAM for user access

#### 24.1.2.5 Other Flash Module Features

- No external high-voltage power supply required for Flash memory program and erase operations
- Interrupt generation on Flash command completion and Flash error detection
- Security mechanism to prevent unauthorized access to the Flash memory

### 24.1.3 Block Diagram

The block diagram of the Flash module is shown in Figure 24-1.

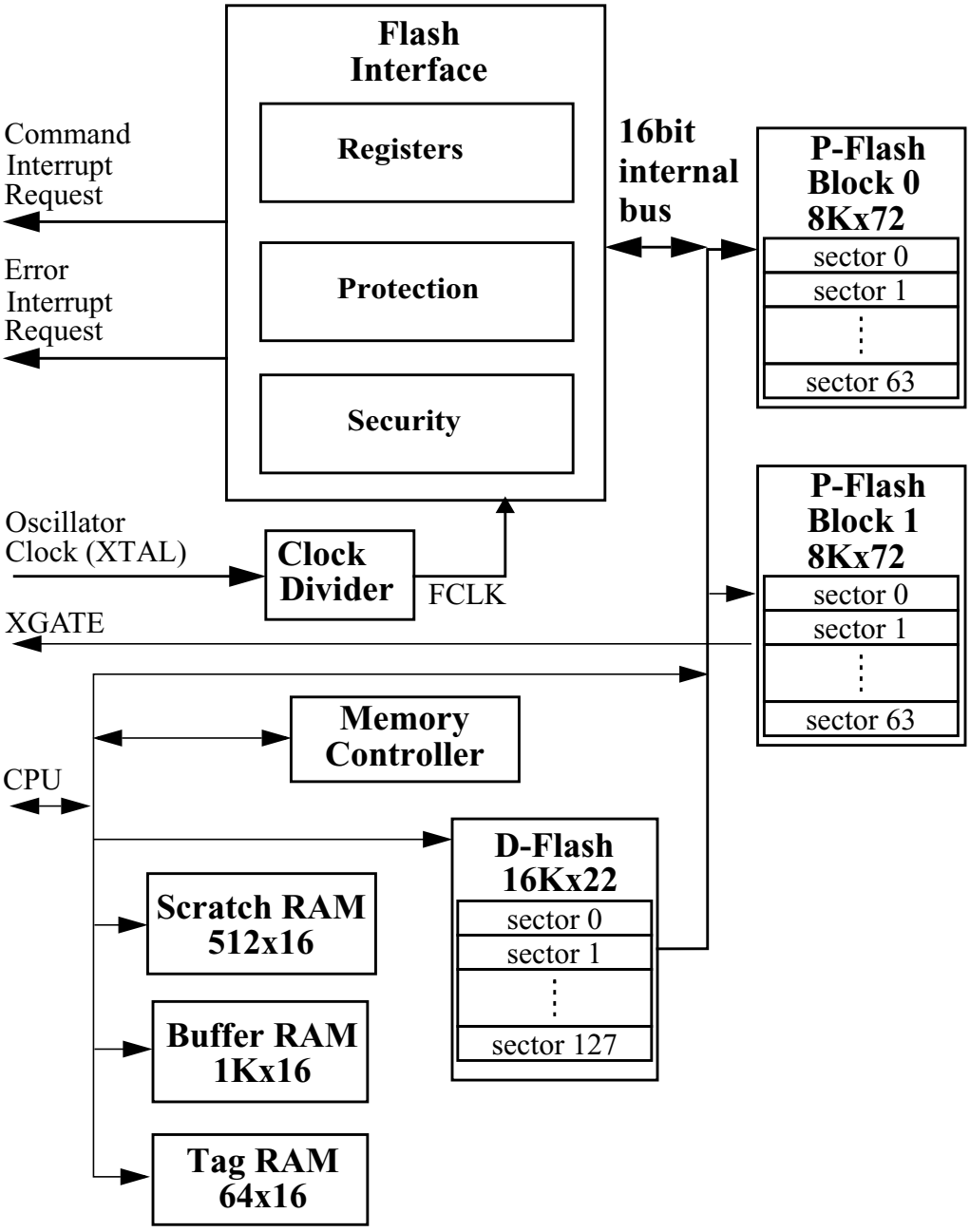



**Figure 24-1. FTM128K2 Block Diagram**

## 24.2 External Signal Description

The Flash module contains no signals that connect off-chip.

## 24.3 Memory Map and Registers

This section describes the memory map and registers for the Flash module. Read data from unimplemented memory space in the Flash module is undefined. Write access to unimplemented or reserved memory space in the Flash module will be ignored by the Flash module.

### 24.3.1 Module Memory Map

The S12X architecture places the P-Flash memory between global addresses 0x78_0000 and 0x7F_FFFF as shown in Table 24-2. The P-Flash memory map is shown in Figure 24-2.

**Table 24-2. P-Flash Memory Addressing**

| Global Address | Size (Bytes) | Description |
|---|---|---|
| 0x7F_0000 – 0x7F_FFFF | 64 K | P-Flash Block 0<br>Contains Flash Configuration Field (see Table 24-3) |
| 0x79_0000 – 0x7E_FFFF | 384 K | No P-Flash Memory |
| 0x78_0000 – 0x78_FFFF | 64 K | P-Flash Block 1 |

The FPROT register, described in Section 24.3.2.9, can be set to protect regions in the Flash memory from accidental program or erase. Three separate memory regions, one growing upward from global address 0x7F_8000 in the Flash memory (called the lower region), one growing downward from global address 0x7F_FFFF in the Flash memory (called the higher region), and the remaining addresses in the Flash memory, can be activated for protection. The Flash memory addresses covered by these protectable regions are shown in the P-Flash memory map. The higher address region is mainly targeted to hold the boot loader code since it covers the vector space. Default protection settings as well as security information that allows the MCU to restrict access to the Flash module are stored in the Flash configuration field as described in Table 24-3.

**Table 24-3. Flash Configuration Field(1)**

| Global Address | Size (Bytes) | Description |
|---|---|---|
| 0x7F_FF00 – 0x7F_FF07 | 8 | Backdoor Comparison Key<br>Refer to Section 24.4.2.11, “Verify Backdoor Access Key Command,” and Section 24.5.1, “Unsecuring the MCU using Backdoor Key Access” |
| 0x7F_FF08 – 0x7F_FF0B(2) | 4 | Reserved |
| 0x7F_FF0C[2] | 1 | P-Flash Protection byte.<br>Refer to Section 24.3.2.9, “P-Flash Protection Register (FPROT)” |
| 0x7F_FF0D[2] | 1 | EEE Protection byte<br>Refer to Section 24.3.2.10, “EEE Protection Register (EPROT)” |
| 0x7F_FF0E[2] | 1 | Flash Nonvolatile byte<br>Refer to Section 24.3.2.14, “Flash Option Register (FOPT)” |

**Table 24-3. Flash Configuration Field[1]**

| Global Address | Size (Bytes) | Description |
|---|---|---|
| 0x7F_FF0F[2] | 1 | Flash Security byte<br>Refer to Section 24.3.2.2, “Flash Security Register (FSEC)” |

1. Older versions may have swapped protection byte addresses
2. 0x7FF08 - 0x7F_FF0F form a Flash phrase and must be programmed in a single command write sequence. Each byte in the 0x7F_FF08 - 0x7F_FF0B reserved field should be programmed to 0xFF.

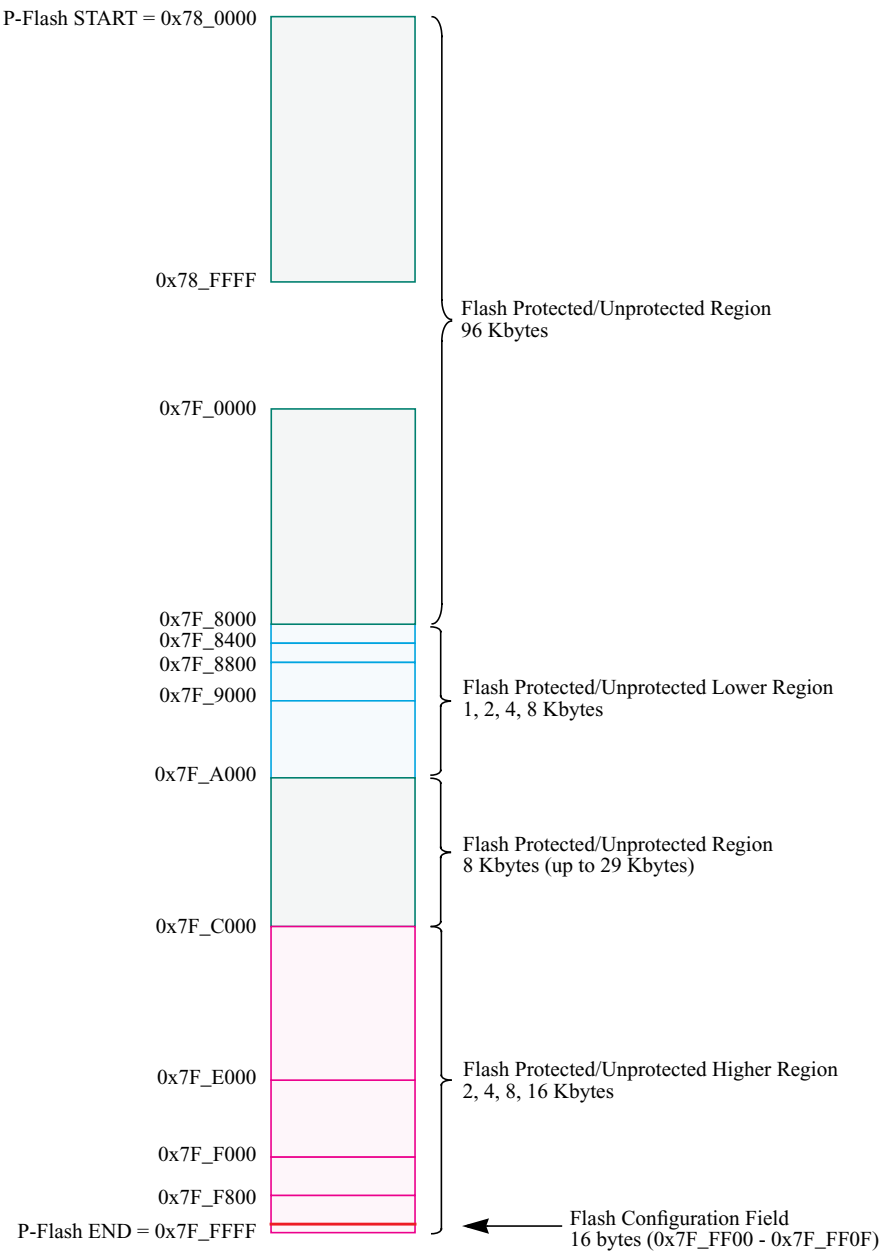



**Figure 24-2. P-Flash Memory Map**

**Table 24-4. Program IFR Fields**

| Global Address (PGMIFRON) | Size (Bytes) | Field Description |
|---|---|---|
| 0x40_0000 – 0x40_0007 | 8 | Device ID |
| 0x40_0008 – 0x40_00E7 | 224 | Reserved |
| 0x40_00E8 – 0x40_00E9 | 2 | Version ID |
| 0x40_00EA – 0x40_00FF | 22 | Reserved |
| 0x40_0100 – 0x40_013F | 64 | Program Once Field<br>Refer to Section 24.4.2.6, “Program Once Command” |
| 0x40_0140 – 0x40_01FF | 192 | Reserved |

**Table 24-5. P-Flash IFR Accessibility**

| Global Address (PGMIFRON) | Size (Bytes) | Accessed From |
|---|---|---|
| 0x40_0000 – 0x40_01FF | 512 | XBUS0 (PBLK0)(1) |
| 0x40_0200 – 0x40_03FF | 512 | Unimplemented |
| 0x40_0400 – 0x40_05FF | 512 | Unimplemented |
| 0x40_0600 – 0x40_07FF | 512 | XBUS1 (PBLK1) |

1. Refer to Table 24-4 for more details.

**Table 24-6. EEE Resource Fields**

| Global Address | Size (Bytes) | Description |
|---|---|---|
| 0x10_0000 – 0x10_7FFF | 32,768 | D-Flash Memory (User and EEE) |
| 0x10_8000 – 0x11_FFFF | 98,304 | Reserved |
| 0x12_0000 – 0x12_007F | 128 | EEE Nonvolatile Information Register (EEEIFRON(1) = 1) |
| 0x12_0080 – 0x12_0FFF | 3,968 | Reserved |
| 0x12_1000 – 0x12_1F7F | 3,968 | Reserved |
| 0x12_1F80 – 0x12_1FFF | 128 | EEE Tag RAM (TMGRAMON[1] = 1) |
| 0x12_2000 – 0x12_3BFF | 7,168 | Reserved |
| 0x12_3C00 – 0x12_3FFF | 1,024 | Memory Controller Scratch RAM (TMGRAMON[1] = 1) |
| 0x12_4000 – 0x12_DFFF | 40,960 | Reserved |
| 0x12_E000 – 0x12_FFFF | 8,192 | Reserved |
| 0x13_0000 – 0x13_F7FF | 63,488 | Reserved |
| 0x13_F800 – 0x13_FFFF | 2,048 | Buffer RAM (User and EEE) |

1. MMCCTL1 register bit

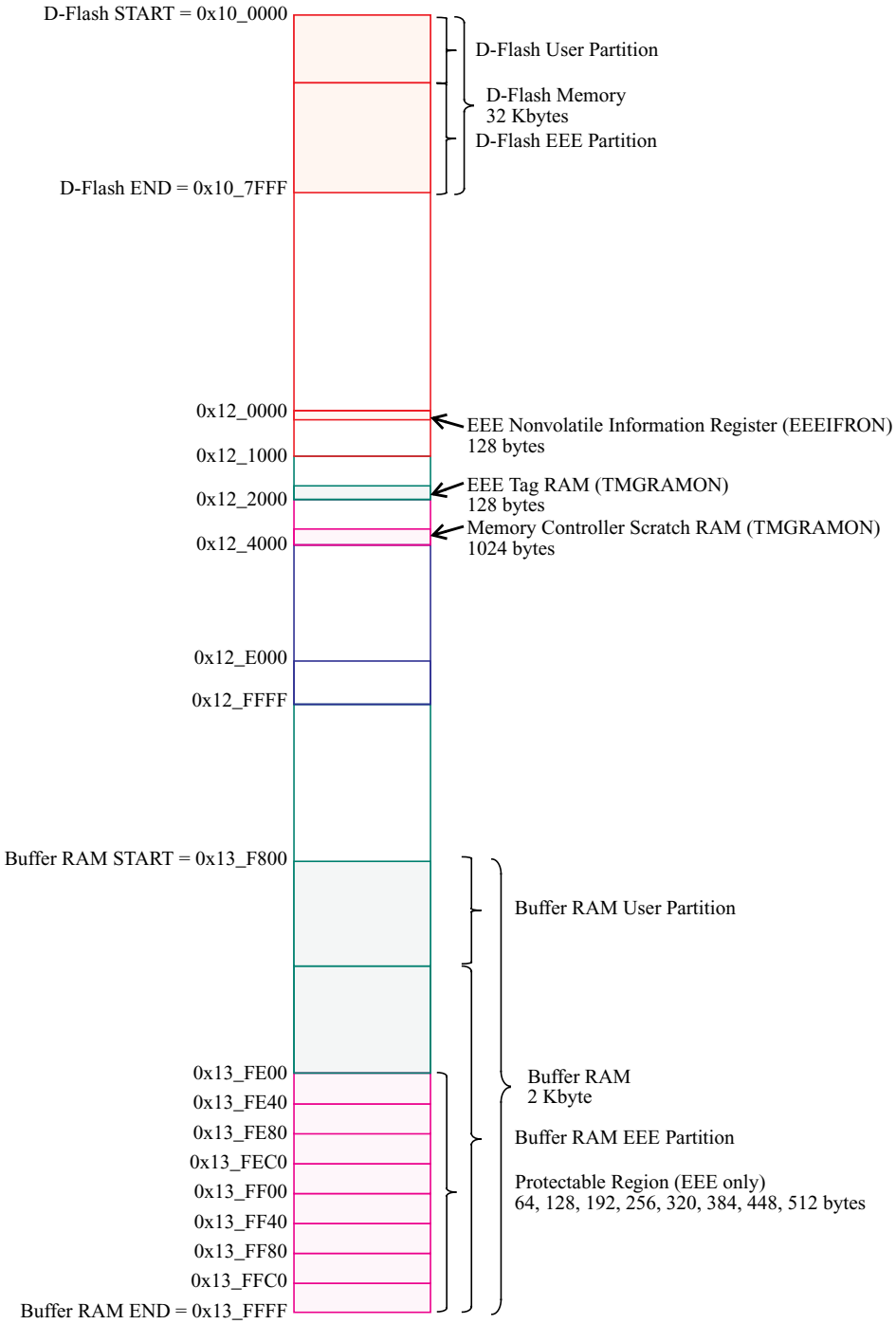



**Figure 24-3. EEE Resource Memory Map**

The Full Partition D-Flash command (see Section 24.4.2.14) is used to program the EEE nonvolatile information register fields where address 0x12_0000 defines the D-Flash partition for user access and address 0x12_0004 defines the buffer RAM partition for EEE operations.

**Table 24-7. EEE Nonvolatile Information Register Fields**

| Global Address (EEEIFRON) | Size (Bytes) | Description |
|---|---|---|
| 0x12_0000 – 0x12_0001 | 2 | D-Flash User Partition (DFPART)<br>Refer to Section 24.4.2.14, “Full Partition D-Flash Command” |
| 0x12_0002 – 0x12_0003 | 2 | D-Flash User Partition (duplicate[1]) |
| 0x12_0004 – 0x12_0005 | 2 | Buffer RAM EEE Partition (ERPART)<br>Refer to Section 24.4.2.14, “Full Partition D-Flash Command” |
| 0x12_0006 – 0x12_0007 | 2 | Buffer RAM EEE Partition (duplicate[1]) |
| 0x12_0008 – 0x12_007F | 120 | Reserved |

1. Duplicate value used if primary value generates a double bit fault when read during the reset sequence.

## 24.3.2 Register Descriptions

The Flash module contains a set of 20 control and status registers located between Flash module base + 0x0000 and 0x0013. A summary of the Flash module registers is given in Figure 24-4 with detailed descriptions in the following subsections.

**CAUTION**

Writes to any Flash register must be avoided while a Flash command is active (CCIF=0) to prevent corruption of Flash register contents and Memory Controller behavior.

| Address & Name | | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|
| 0x0000 FCLKDIV | R | FDIVLD | FDIV6 | FDIV5 | FDIV4 | FDIV3 | FDIV2 | FDIV1 | FDIV0 |
| | W | | | | | | | | |
| 0x0001 FSEC | R | KEYEN1 | KEYEN0 | RNV5 | RNV4 | RNV3 | RNV2 | SEC1 | SEC0 |
| | W | | | | | | | | |
| 0x0002 FCCOBIX | R | 0 | 0 | 0 | 0 | 0 | CCOBIX2 | CCOBIX1 | CCOBIX0 |
| | W | | | | | | | | |
| 0x0003 FECCRIX | R | 0 | 0 | 0 | 0 | 0 | ECCRIX2 | ECCRIX1 | ECCRIX0 |
| | W | | | | | | | | |
| 0x0004 FCNFG | R | CCIE | 0 | 0 | IGNSF | 0 | 0 | FDFD | FSFD |
| | W | | | | | | | | |

**Figure 24-4. FTM128K2 Register Summary**

| Address & Name | | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|
| 0x0005 FERCNFG | R | ERSERIE | PGMERIE | 0 | EPVIOLIE | ERSVIE1 | ERSVIE0 | DFDIE | SFDIE |
| | W | | | | | | | | |
| 0x0006 FSTAT | R | CCIF | 0 | ACCERR | FPVIOL | MGBUSY | RSVD | MGSTAT1 | MGSTAT0 |
| | W | | | | | | | | |
| 0x0007 FERSTAT | R | ERSERIF | PGMERIF | 0 | EPVIOLIF | ERSVIF1 | ERSVIF0 | DFDIF | SFDIF |
| | W | | | | | | | | |
| 0x0008 FPROT | R | FPOPEN | RNV6 | FPHDIS | FPHS1 | FPHS0 | FPLDIS | FPLS1 | FPLS0 |
| | W | | | | | | | | |
| 0x0009 EPROT | R | EPOPEN | RNV6 | RNV5 | RNV4 | EPDIS | EPS2 | EPS1 | EPS0 |
| | W | | | | | | | | |
| 0x000A FCCOBHI | R | CCOB15 | CCOB14 | CCOB13 | CCOB12 | CCOB11 | CCOB10 | CCOB9 | CCOB8 |
| | W | | | | | | | | |
| 0x000B FCCOBLO | R | CCOB7 | CCOB6 | CCOB5 | CCOB4 | CCOB3 | CCOB2 | CCOB1 | CCOB0 |
| | W | | | | | | | | |
| 0x000C ETAGHI | R | ETAG15 | ETAG14 | ETAG13 | ETAG12 | ETAG11 | ETAG10 | ETAG9 | ETAG8 |
| | W | | | | | | | | |
| 0x000D ETAGLO | R | ETAG7 | ETAG6 | ETAG5 | ETAG4 | ETAG3 | ETAG2 | ETAG1 | ETAG0 |
| | W | | | | | | | | |
| 0x000E FECCRHI | R | ECCR15 | ECCR14 | ECCR13 | ECCR12 | ECCR11 | ECCR10 | ECCR9 | ECCR8 |
| | W | | | | | | | | |
| 0x000F FECCRLO | R | ECCR7 | ECCR6 | ECCR5 | ECCR4 | ECCR3 | ECCR2 | ECCR1 | ECCR0 |
| | W | | | | | | | | |
| 0x0010 FOPT | R | NV7 | NV6 | NV5 | NV4 | NV3 | NV2 | NV1 | NV0 |
| | W | | | | | | | | |
| 0x0011 FRSV0 | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | W | | | | | | | | |
| 0x0012 FRSV1 | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | W | | | | | | | | |

**Figure 24-4. FTM128K2 Register Summary (continued)**

| Address & Name | | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|
| 0x0013 FRSV2 | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | W | | | | | | | | |

= Unimplemented or Reserved

**Figure 24-4. FTM128K2 Register Summary (continued)**

### 24.3.2.1 Flash Clock Divider Register (FCLKDIV)

The FCLKDIV register is used to control timed events in program and erase algorithms.

Offset Module Base + 0x0000

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | FDIVLD | FDIV[6:0] | | | | | | |
| W | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented or Reserved

**Figure 24-5. Flash Clock Divider Register (FCLKDIV)**

All bits in the FCLKDIV register are readable, bits 6–0 are write once and bit 7 is not writable.

**Table 24-8. FCLKDIV Field Descriptions**

| Field | Description |
|---|---|
| 7<br>FDIVLD | **Clock Divider Loaded**<br>0 FCLKDIV register has not been written<br>1 FCLKDIV register has been written since the last reset |
| 6–0<br>FDIV[6:0] | **Clock Divider Bits** — FDIV[6:0] must be set to effectively divide OSCCLK down to generate an internal Flash clock, FCLK, with a target frequency of 1 MHz for use by the Flash module to control timed events during program and erase algorithms. Table 24-9 shows recommended values for FDIV[6:0] based on OSCCLK frequency. Please refer to Section 24.4.1, “Flash Command Operations,” for more information. |

**CAUTION**

The FCLKDIV register should never be written while a Flash command is executing (CCIF=0). The FCLKDIV register is writable during the Flash reset sequence even though CCIF is clear.

**Table 24-9. FDIV vs OSCCLK Frequency**

| OSCCLK Frequency (MHz) | | FDIV[6:0] | OSCCLK Frequency (MHz) | | FDIV[6:0] | OSCCLK Frequency (MHz) | | FDIV[6:0] |
|---|---|---|---|---|---|---|---|---|
| MIN(1) | MAX(2) | | MIN1 | MAX2 | | MIN1 | MAX2 | |
| | | | 33.60 | 34.65 | 0x20 | 67.20 | 68.25 | 0x40 |
| 1.60 | 2.10 | 0x01 | 34.65 | 35.70 | 0x21 | 68.25 | 69.30 | 0x41 |
| 2.40 | 3.15 | 0x02 | 35.70 | 36.75 | 0x22 | 69.30 | 70.35 | 0x42 |
| 3.20 | 4.20 | 0x03 | 36.75 | 37.80 | 0x23 | 70.35 | 71.40 | 0x43 |
| 4.20 | 5.25 | 0x04 | 37.80 | 38.85 | 0x24 | 71.40 | 72.45 | 0x44 |
| 5.25 | 6.30 | 0x05 | 38.85 | 39.90 | 0x25 | 72.45 | 73.50 | 0x45 |
| 6.30 | 7.35 | 0x06 | 39.90 | 40.95 | 0x26 | 73.50 | 74.55 | 0x46 |
| 7.35 | 8.40 | 0x07 | 40.95 | 42.00 | 0x27 | 74.55 | 75.60 | 0x47 |
| 8.40 | 9.45 | 0x08 | 42.00 | 43.05 | 0x28 | 75.60 | 76.65 | 0x48 |
| 9.45 | 10.50 | 0x09 | 43.05 | 44.10 | 0x29 | 76.65 | 77.70 | 0x49 |
| 10.50 | 11.55 | 0x0A | 44.10 | 45.15 | 0x2A | 77.70 | 78.75 | 0x4A |
| 11.55 | 12.60 | 0x0B | 45.15 | 46.20 | 0x2B | 78.75 | 79.80 | 0x4B |
| 12.60 | 13.65 | 0x0C | 46.20 | 47.25 | 0x2C | 79.80 | 80.85 | 0x4C |
| 13.65 | 14.70 | 0x0D | 47.25 | 48.30 | 0x2D | 80.85 | 81.90 | 0x4D |
| 14.70 | 15.75 | 0x0E | 48.30 | 49.35 | 0x2E | 81.90 | 82.95 | 0x4E |
| 15.75 | 16.80 | 0x0F | 49.35 | 50.40 | 0x2F | 82.95 | 84.00 | 0x4F |
| 16.80 | 17.85 | 0x10 | 50.40 | 51.45 | 0x30 | 84.00 | 85.05 | 0x50 |
| 17.85 | 18.90 | 0x11 | 51.45 | 52.50 | 0x31 | 85.05 | 86.10 | 0x51 |
| 18.90 | 19.95 | 0x12 | 52.50 | 53.55 | 0x32 | 86.10 | 87.15 | 0x52 |
| 19.95 | 21.00 | 0x13 | 53.55 | 54.60 | 0x33 | 87.15 | 88.20 | 0x53 |
| 21.00 | 22.05 | 0x14 | 54.60 | 55.65 | 0x34 | 88.20 | 89.25 | 0x54 |
| 22.05 | 23.10 | 0x15 | 55.65 | 56.70 | 0x35 | 89.25 | 90.30 | 0x55 |
| 23.10 | 24.15 | 0x16 | 56.70 | 57.75 | 0x36 | 90.30 | 91.35 | 0x56 |
| 24.15 | 25.20 | 0x17 | 57.75 | 58.80 | 0x37 | 91.35 | 92.40 | 0x57 |
| 25.20 | 26.25 | 0x18 | 58.80 | 59.85 | 0x38 | 92.40 | 93.45 | 0x58 |
| 26.25 | 27.30 | 0x19 | 59.85 | 60.90 | 0x39 | 93.45 | 94.50 | 0x59 |
| 27.30 | 28.35 | 0x1A | 60.90 | 61.95 | 0x3A | 94.50 | 95.55 | 0x5A |
| 28.35 | 29.40 | 0x1B | 61.95 | 63.00 | 0x3B | 95.55 | 96.60 | 0x5B |
| 29.40 | 30.45 | 0x1C | 63.00 | 64.05 | 0x3C | 96.60 | 97.65 | 0x5C |
| 30.45 | 31.50 | 0x1D | 64.05 | 65.10 | 0x3D | 97.65 | 98.70 | 0x5D |
| 31.50 | 32.55 | 0x1E | 65.10 | 66.15 | 0x3E | 98.70 | 99.75 | 0x5E |
| 32.55 | 33.60 | 0x1F | 66.15 | 67.20 | 0x3F | 99.75 | 100.80 | 0x5F |

1. FDIV shown generates an FCLK frequency of >0.8 MHz

2. FDIV shown generates an FCLK frequency of 1.05 MHz

### 24.3.2.2 Flash Security Register (FSEC)

The FSEC register holds all bits associated with the security of the MCU and Flash module.

Offset Module Base + 0x0001

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | KEYEN[1:0] | | RNV[5:2] | | | | SEC[1:0] | |
| W | | | | | | | | |
| Reset | F | F | F | F | F | F | F | F |

= Unimplemented or Reserved

**Figure 24-6. Flash Security Register (FSEC)**

All bits in the FSEC register are readable but not writable.

During the reset sequence, the FSEC register is loaded with the contents of the Flash security byte in the Flash configuration field at global address 0x7F_FF0F located in P-Flash memory (see Table 24-3) as indicated by reset condition F in Figure 24-6. If a double bit fault is detected while reading the P-Flash phrase containing the Flash security byte during the reset sequence, all bits in the FSEC register will be set to leave the Flash module in a secured state with backdoor key access disabled.

**Table 24-10. FSEC Field Descriptions**

| Field | Description |
|---|---|
| 7–6<br>KEYEN[1:0] | **Backdoor Key Security Enable Bits** — The KEYEN[1:0] bits define the enabling of backdoor key access to the Flash module as shown in Table 24-11. |
| 5–2<br>RNV[5:2} | **Reserved Nonvolatile Bits** — The RNV bits should remain in the erased state for future enhancements. |
| 1–0<br>SEC[1:0] | **Flash Security Bits** — The SEC[1:0] bits define the security state of the MCU as shown in Table 24-12. If the Flash module is unsecured using backdoor key access, the SEC bits are forced to 10. |

**Table 24-11. Flash KEYEN States**

| KEYEN[1:0] | Status of Backdoor Key Access |
|---|---|
| 00 | DISABLED |
| 01 | DISABLED(1) |
| 10 | ENABLED |
| 11 | DISABLED |

1. Preferred KEYEN state to disable backdoor key access.

**Table 24-12. Flash Security States**

| SEC[1:0] | Status of Security |
|---|---|
| 00 | SECURED |
| 01 | SECURED(1) |
| 10 | UNSECURED |
| 11 | SECURED |

1. Preferred SEC state to set MCU to secured state.

The security function in the Flash module is described in Section 24.5.

### 24.3.2.3 Flash CCOB Index Register (FCCOBIX)

The FCCOBIX register is used to index the FCCOB register for Flash memory operations.

Offset Module Base + 0x0002

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | 0 | 0 | 0 | CCOBIX[2:0] | | |
| W | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented or Reserved

**Figure 24-7. FCCOB Index Register (FCCOBIX)**

CCOBIX bits are readable and writable while remaining bits read 0 and are not writable.

**Table 24-13. FCCOBIX Field Descriptions**

| Field | Description |
|---|---|
| 2–0 CCOBIX[1:0] | **Common Command Register Index**— The CCOBIX bits are used to select which word of the FCCOB register array is being read or written to. See Section 24.3.2.11, “Flash Common Command Object Register (FCCOB),” for more details. |

### 24.3.2.4 Flash ECCR Index Register (FECCRIX)

The FECCRIX register is used to index the FECCR register for ECC fault reporting.

Offset Module Base + 0x0003

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | 0 | 0 | 0 | ECCRIX[2:0] | | |
| W | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented or Reserved

**Figure 24-8. FECCR Index Register (FECCRIX)**

ECCRIX bits are readable and writable while remaining bits read 0 and are not writable.

**Table 24-14. FECCRIX Field Descriptions**

| Field | Description |
|---|---|
| 2-0<br>ECCRIX[2:0] | **ECC Error Register Index**— The ECCRIX bits are used to select which word of the FECCR register array is being read. See Section 24.3.2.13, “Flash ECC Error Results Register (FECCR),” for more details. |

### 24.3.2.5 Flash Configuration Register (FCNFG)

The FCNFG register enables the Flash command complete interrupt and forces ECC faults on Flash array read access from the CPU or XGATE.

Offset Module Base + 0x0004

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | CCIE | 0 | 0 | IGNSF | 0 | 0 | FDFD | FSFD |
| W | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented or Reserved

**Figure 24-9. Flash Configuration Register (FCNFG)**

CCIE, IGNSF, FDFD, and FSFD bits are readable and writable while remaining bits read 0 and are not writable.

**Table 24-15. FCNFG Field Descriptions**

| Field | Description |
|---|---|
| 7<br>CCIE | **Command Complete Interrupt Enable** — The CCIE bit controls interrupt generation when a Flash command has completed.<br>0 Command complete interrupt disabled<br>1 An interrupt will be requested whenever the CCIF flag in the FSTAT register is set (see Section 24.3.2.7) |
| 4<br>IGNSF | **Ignore Single Bit Fault** — The IGNSF controls single bit fault reporting in the FERSTAT register (see Section 24.3.2.8).<br>0 All single bit faults detected during array reads are reported<br>1 Single bit faults detected during array reads are not reported and the single bit fault interrupt will not be generated |

**Table 24-15. FCNFG Field Descriptions (continued)**

| Field | Description |
|---|---|
| 1<br>FDFD | **Force Double Bit Fault Detect** — The FDFD bit allows the user to simulate a double bit fault during Flash array read operations and check the associated interrupt routine. The FDFD bit is cleared by writing a 0 to FDFD. The FECCR registers will not be updated during the Flash array read operation with FDFD set unless an actual double bit fault is detected.<br>0 Flash array read operations will set the DFDIF flag in the FERSTAT register only if a double bit fault is detected<br>1 Any Flash array read operation will force the DFDIF flag in the FERSTAT register to be set (see Section 24.3.2.7) and an interrupt will be generated as long as the DFDIE interrupt enable in the FERCNFG register is set (see Section 24.3.2.6) |
| 0<br>FSFD | **Force Single Bit Fault Detect** — The FSFD bit allows the user to simulate a single bit fault during Flash array read operations and check the associated interrupt routine. The FSFD bit is cleared by writing a 0 to FSFD. The FECCR registers will not be updated during the Flash array read operation with FSFD set unless an actual single bit fault is detected.<br>0 Flash array read operations will set the SFDIF flag in the FERSTAT register only if a single bit fault is detected<br>1 Flash array read operation will force the SFDIF flag in the FERSTAT register to be set (see Section 24.3.2.7) and an interrupt will be generated as long as the SFDIE interrupt enable in the FERCNFG register is set (see Section 24.3.2.6) |

### 24.3.2.6 Flash Error Configuration Register (FERCNFG)

The FERCNFG register enables the Flash error interrupts for the FERSTAT flags.

Offset Module Base + 0x0005

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | ERSERIE | PGMERIE | 0 | EPVIOLIE | ERSVIE1 | ERSVIE0 | DFDIE | SFDIE |
| W | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented or Reserved

**Figure 24-10. Flash Error Configuration Register (FERCNFG)**

All assigned bits in the FERCNFG register are readable and writable.

**Table 24-16. FERCNFG Field Descriptions**

| Field | Description |
|---|---|
| 7<br>ERSERIE | **EEE Erase Error Interrupt Enable** — The ERSERIE bit controls interrupt generation when a failure is detected during an EEE erase operation.<br>0 ERSERIF interrupt disabled<br>1 An interrupt will be requested whenever the ERSERIF flag is set (see Section 24.3.2.8) |
| 6<br>PGMERIE | **EEE Program Error Interrupt Enable** — The PGMERIE bit controls interrupt generation when a failure is detected during an EEE program operation.<br>0 PGMERIF interrupt disabled<br>1 An interrupt will be requested whenever the PGMERIF flag is set (see Section 24.3.2.8) |
| 4<br>EPVIOLIE | **EEE Protection Violation Interrupt Enable** — The EPVIOLIE bit controls interrupt generation when a protection violation is detected during a write to the buffer RAM EEE partition.<br>0 EPVIOLIF interrupt disabled<br>1 An interrupt will be requested whenever the EPVIOLIF flag is set (see Section 24.3.2.8) |

**Table 24-16. FERCNFG Field Descriptions (continued)**

| Field | Description |
|---|---|
| 3<br>ERSVIE1 | **EEE Error Type 1 Interrupt Enable** — The ERSVIE1 bit controls interrupt generation when a change state error is detected during an EEE operation.<br>0 ERSVIF1 interrupt disabled<br>1 An interrupt will be requested whenever the ERSVIF1 flag is set (see Section 24.3.2.8) |
| 2<br>ERSVIE0 | **EEE Error Type 0 Interrupt Enable** — The ERSVIE0 bit controls interrupt generation when a sector format error is detected during an EEE operation.<br>0 ERSVIF0 interrupt disabled<br>1 An interrupt will be requested whenever the ERSVIF0 flag is set (see Section 24.3.2.8) |
| 1<br>DFDIE | **Double Bit Fault Detect Interrupt Enable** — The DFDIE bit controls interrupt generation when a double bit fault is detected during a Flash block read operation.<br>0 DFDIF interrupt disabled<br>1 An interrupt will be requested whenever the DFDIF flag is set (see Section 24.3.2.8) |
| 0<br>SFDIE | **Single Bit Fault Detect Interrupt Enable** — The SFDIE bit controls interrupt generation when a single bit fault is detected during a Flash block read operation.<br>0 SFDIF interrupt disabled whenever the SFDIF flag is set (see Section 24.3.2.8)<br>1 An interrupt will be requested whenever the SFDIF flag is set (see Section 24.3.2.8) |

### 24.3.2.7 Flash Status Register (FSTAT)

The FSTAT register reports the operational status of the Flash module.

Offset Module Base + 0x0006

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | CCIF | 0 | ACCERR | FPVIOL | MGBUSY | RSVD | MGSTAT[1:0] | |
| W | | | | | | | | |
| Reset | 1 | 0 | 0 | 0 | 0 | 0 | 0(1) | 01 |

= Unimplemented or Reserved

**Figure 24-11. Flash Status Register (FSTAT)**

1. Reset value can deviate from the value shown if a double bit fault is detected during the reset sequence (see Section 24.6).

CCIF, ACCERR, and FPVIOL bits are readable and writable, MGBUSY and MGSTAT bits are readable but not writable, while remaining bits read 0 and are not writable.

**Table 24-17. FSTAT Field Descriptions**

| Field | Description |
|---|---|
| 7<br>CCIF | **Command Complete Interrupt Flag** — The CCIF flag indicates that a Flash command has completed. The CCIF flag is cleared by writing a 1 to CCIF to launch a command and CCIF will stay low until command completion or command violation.<br>0 Flash command in progress<br>1 Flash command has completed |
| 5<br>ACCERR | **Flash Access Error Flag** — The ACCERR bit indicates an illegal access has occurred to the Flash memory caused by either a violation of the command write sequence (see Section 24.4.1.2) or issuing an illegal Flash command or when errors are encountered while initializing the EEE buffer ram during the reset sequence. While ACCERR is set, the CCIF flag cannot be cleared to launch a command. The ACCERR bit is cleared by writing a 1 to ACCERR. Writing a 0 to the ACCERR bit has no effect on ACCERR.<br>0 No access error detected<br>1 Access error detected |
| 4<br>FPVIOL | **Flash Protection Violation Flag** —The FPVIOL bit indicates an attempt was made to program or erase an address in a protected area of P-Flash memory during a command write sequence. The FPVIOL bit is cleared by writing a 1 to FPVIOL. Writing a 0 to the FPVIOL bit has no effect on FPVIOL. While FPVIOL is set, it is not possible to launch a command or start a command write sequence.<br>0 No protection violation detected<br>1 Protection violation detected |
| 3<br>MGBUSY | **Memory Controller Busy Flag** — The MGBUSY flag reflects the active state of the Memory Controller.<br>0 Memory Controller is idle<br>1 Memory Controller is busy executing a Flash command (CCIF = 0) or is handling internal EEE operations |
| 2<br>RSVD | **Reserved Bit** — This bit is reserved and always reads 0. |
| 1–0<br>MGSTAT[1:0] | **Memory Controller Command Completion Status Flag** — One or more MGSTAT flag bits are set if an error is detected during execution of a Flash command or during the Flash reset sequence. See Section 24.4.2, “Flash Command Description,” and Section 24.6, “Initialization” for details. |

### 24.3.2.8 Flash Error Status Register (FERSTAT)

The FERSTAT register reflects the error status of internal Flash operations.

Offset Module Base + 0x0007

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | ERSERIF | PGMERIF | 0 | EPVIOLIF | ERSVIF1 | ERSVIF0 | DFDIF | SFDIF |
| W | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented or Reserved

**Figure 24-12. Flash Error Status Register (FERSTAT)**

All flags in the FERSTAT register are readable and only writable to clear the flag.

**Table 24-18. FERSTAT Field Descriptions**

| Field | Description |
|---|---|
| 7<br>ERSERIF | **EEE Erase Error Interrupt Flag** — The setting of the ERSERIF flag occurs due to an error in a Flash erase command that resulted in the erase operation not being successful during EEE operations. The ERSERIF flag is cleared by writing a 1 to ERSERIF. Writing a 0 to the ERSERIF flag has no effect on ERSERIF. While ERSERIF is set, it is possible to write to the buffer RAM EEE partition but the data written will not be transferred to the D-Flash EEE partition.<br>0 Erase command successfully completed on the D-Flash EEE partition<br>1 Erase command failed on the D-Flash EEE partition |
| 6<br>PGMERIF | **EEE Program Error Interrupt Flag** — The setting of the PGMERIF flag occurs due to an error in a Flash program command that resulted in the program operation not being successful during EEE operations. The PGMERIF flag is cleared by writing a 1 to PGMERIF. Writing a 0 to the PGMERIF flag has no effect on PGMERIF. While PGMERIF is set, it is possible to write to the buffer RAM EEE partition but the data written will not be transferred to the D-Flash EEE partition.<br>0 Program command successfully completed on the D-Flash EEE partition<br>1 Program command failed on the D-Flash EEE partition |
| 4<br>EPVIOLIF | **EEE Protection Violation Interrupt Flag** —The setting of the EPVIOLIF flag indicates an attempt was made to write to a protected area of the buffer RAM EEE partition. The EPVIOLIF flag is cleared by writing a 1 to EPVIOLIF. Writing a 0 to the EPVIOLIF flag has no effect on EPVIOLIF. While EPVIOLIF is set, it is possible to write to the buffer RAM EEE partition as long as the address written to is not in a protected area.<br>0 No EEE protection violation<br>1 EEE protection violation detected |
| 3<br>ERSVIF1 | **EEE Error Interrupt 1 Flag** —The setting of the ERSVIF1 flag indicates that the memory controller was unable to change the state of a D-Flash EEE sector. The ERSVIF1 flag is cleared by writing a 1 to ERSVIF1. Writing a 0 to the ERSVIF1 flag has no effect on ERSVIF1. While ERSVIF1 is set, it is possible to write to the buffer RAM EEE partition but the data written will not be transferred to the D-Flash EEE partition.<br>0 No EEE sector state change error detected<br>1 EEE sector state change error detected |
| 2<br>ERSVIF0 | **EEE Error Interrupt 0 Flag** —The setting of the ERSVIF0 flag indicates that the memory controller was unable to format a D-Flash EEE sector for EEE use. The ERSVIF0 flag is cleared by writing a 1 to ERSVIF0. Writing a 0 to the ERSVIF0 flag has no effect on ERSVIF0. While ERSVIF0 is set, it is possible to write to the buffer RAM EEE partition but the data written will not be transferred to the D-Flash EEE partition.<br>0 No EEE sector format error detected<br>1 EEE sector format error detected |
| 1<br>DFDIF | **Double Bit Fault Detect Interrupt Flag** — The setting of the DFDIF flag indicates that a double bit fault was detected in the stored parity and data bits during a Flash array read operation or that a Flash array read operation was attempted on a Flash block that was under a Flash command operation. The DFDIF flag is cleared by writing a 1 to DFDIF. Writing a 0 to DFDIF has no effect on DFDIF.<br>0 No double bit fault detected<br>1 Double bit fault detected or an invalid Flash array read operation attempted |
| 0<br>SFDIF | **Single Bit Fault Detect Interrupt Flag** — With the IGNSF bit in the FCNFG register clear, the SFDIF flag indicates that a single bit fault was detected in the stored parity and data bits during a Flash array read operation or that a Flash array read operation was attempted on a Flash block that was under a Flash command operation. The SFDIF flag is cleared by writing a 1 to SFDIF. Writing a 0 to SFDIF has no effect on SFDIF.<br>0 No single bit fault detected<br>1 Single bit fault detected and corrected or an invalid Flash array read operation attempted |

### 24.3.2.9 P-Flash Protection Register (FPROT)

The FPROT register defines which P-Flash sectors are protected against program and erase operations.

Offset Module Base + 0x0008

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | FPOPEN | RNV6 | FPHDIS | FPHS[1:0] | | FPLDIS | FPLS[1:0] | |
| W | | | | | | | | |
| Reset | F | F | F | F | F | F | F | F |

= Unimplemented or Reserved

**Figure 24-13. Flash Protection Register (FPROT)**

The (unreserved) bits of the FPROT register are writable with the restriction that the size of the protected region can only be increased (see Section 24.3.2.9.1, “P-Flash Protection Restrictions,” and Table 24-23).

During the reset sequence, the FPROT register is loaded with the contents of the P-Flash protection byte in the Flash configuration field at global address 0x7F_FF0C located in P-Flash memory (see Table 24-3) as indicated by reset condition ‘F’ in Figure 24-13. To change the P-Flash protection that will be loaded during the reset sequence, the upper sector of the P-Flash memory must be unprotected, then the P-Flash protection byte must be reprogrammed. If a double bit fault is detected while reading the P-Flash phrase containing the P-Flash protection byte during the reset sequence, the FPOPEN bit will be cleared and remaining bits in the FPROT register will be set to leave the P-Flash memory fully protected.

Trying to alter data in any protected area in the P-Flash memory will result in a protection violation error and the FPVIOL bit will be set in the FSTAT register. The block erase of a P-Flash block is not possible if any of the P-Flash sectors contained in the same P-Flash block are protected.

**Table 24-19. FPROT Field Descriptions**

| Field | Description |
|---|---|
| 7<br>FPOPEN | **Flash Protection Operation Enable** — The FPOPEN bit determines the protection function for program or erase operations as shown in Table 24-20 for the P-Flash block.<br>0 When FPOPEN is clear, the FPHDIS and FPLDIS bits define unprotected address ranges as specified by the corresponding FPHS and FPLS bits<br>1 When FPOPEN is set, the FPHDIS and FPLDIS bits enable protection for the address range specified by the corresponding FPHS and FPLS bits |
| 6<br>RNV[6] | **Reserved Nonvolatile Bit** — The RNV bit should remain in the erased state for future enhancements. |
| 5<br>FPHDIS | **Flash Protection Higher Address Range Disable** — The FPHDIS bit determines whether there is a protected/unprotected area in a specific region of the P-Flash memory ending with global address 0x7F_FFFF.<br>0 Protection/Unprotection enabled<br>1 Protection/Unprotection disabled |
| 4–3<br>FPHS[1:0] | **Flash Protection Higher Address Size** — The FPHS bits determine the size of the protected/unprotected area in P-Flash memory as shown inTable 24-21. The FPHS bits can only be written to while the FPHDIS bit is set. |
| 2<br>FPLDIS | **Flash Protection Lower Address Range Disable** — The FPLDIS bit determines whether there is a protected/unprotected area in a specific region of the P-Flash memory beginning with global address 0x7F_8000.<br>0 Protection/Unprotection enabled<br>1 Protection/Unprotection disabled |
| 1–0<br>FPLS[1:0] | **Flash Protection Lower Address Size** — The FPLS bits determine the size of the protected/unprotected area in P-Flash memory as shown in Table 24-22. The FPLS bits can only be written to while the FPLDIS bit is set. |

**Table 24-20. P-Flash Protection Function**

| FPOPEN | FPHDIS | FPLDIS | Function[1] |
|---|---|---|---|
| 1 | 1 | 1 | No P-Flash Protection |
| 1 | 1 | 0 | Protected Low Range |
| 1 | 0 | 1 | Protected High Range |
| 1 | 0 | 0 | Protected High and Low Ranges |
| 0 | 1 | 1 | Full P-Flash Memory Protected |
| 0 | 1 | 0 | Unprotected Low Range |
| 0 | 0 | 1 | Unprotected High Range |
| 0 | 0 | 0 | Unprotected High and Low Ranges |

1. For range sizes, refer to Table 24-21 and Table 24-22.

**Table 24-21. P-Flash Protection Higher Address Range**

| FPHS[1:0] | Global Address Range | Protected Size |
|---|---|---|
| 00 | 0x7F_F800–0x7F_FFFF | 2 Kbytes |
| 01 | 0x7F_F000–0x7F_FFFF | 4 Kbytes |
| 10 | 0x7F_E000–0x7F_FFFF | 8 Kbytes |
| 11 | 0x7F_C000–0x7F_FFFF | 16 Kbytes |

**Table 24-22. P-Flash Protection Lower Address Range**

| FPLS[1:0] | Global Address Range | Protected Size |
|---|---|---|
| 00 | 0x7F_8000–0x7F_83FF | 1 Kbyte |
| 01 | 0x7F_8000–0x7F_87FF | 2 Kbytes |
| 10 | 0x7F_8000–0x7F_8FFF | 4 Kbytes |
| 11 | 0x7F_8000–0x7F_9FFF | 8 Kbytes |

All possible P-Flash protection scenarios are shown in Figure 24-14. Although the protection scheme is loaded from the Flash memory at global address 0x7F_FF0C during the reset sequence, it can be changed by the user. The P-Flash protection scheme can be used by applications requiring reprogramming in single chip mode while providing as much protection as possible if reprogramming is not required.

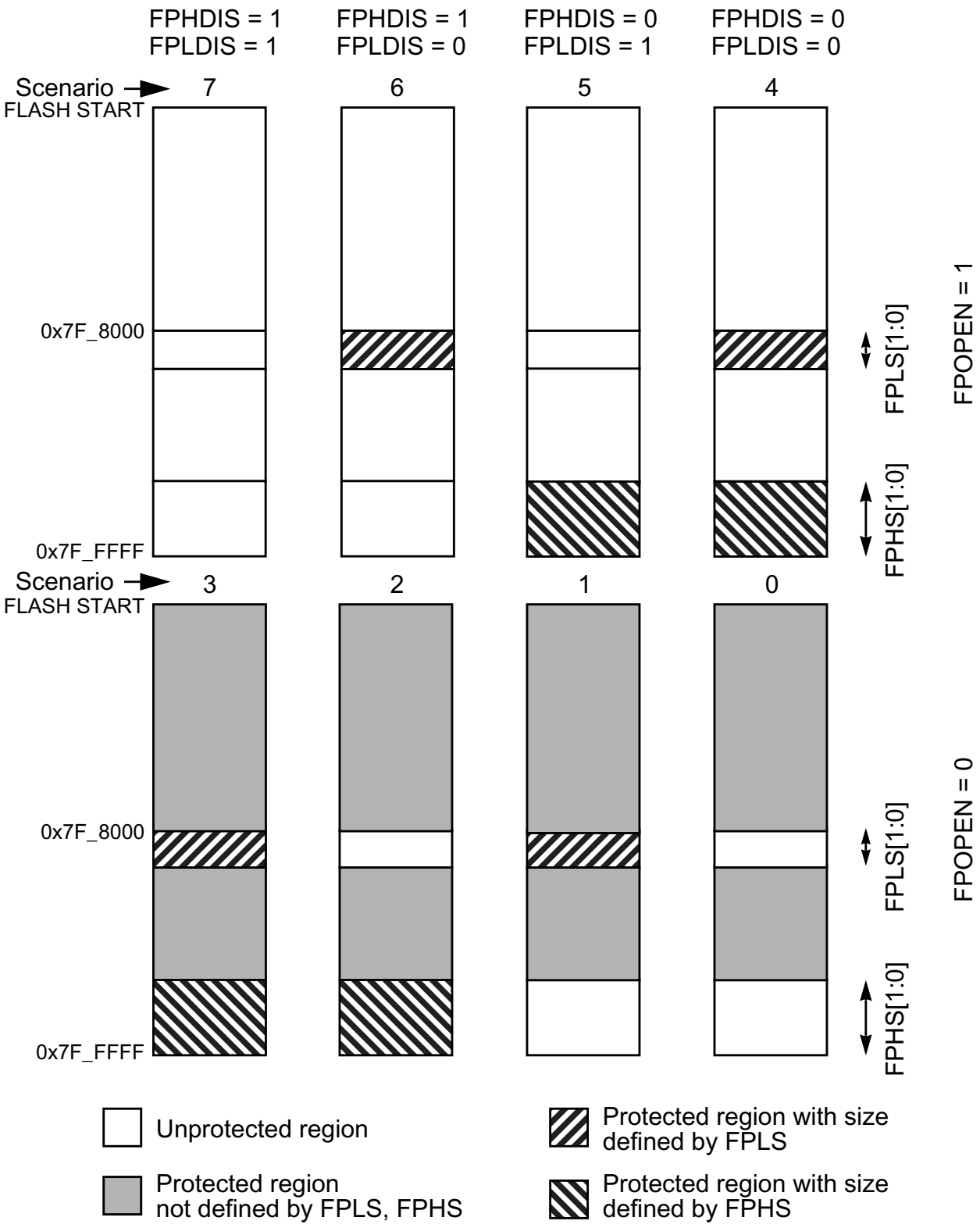



**Figure 24-14. P-Flash Protection Scenarios**

#### 24.3.2.9.1 P-Flash Protection Restrictions

The general guideline is that P-Flash protection can only be added and not removed. Table 24-23 specifies all valid transitions between P-Flash protection scenarios. Any attempt to write an invalid scenario to the FPROT register will be ignored. The contents of the FPROT register reflect the active protection scenario. See the FPHS and FPLS bit descriptions for additional restrictions.

**Table 24-23. P-Flash Protection Scenario Transitions**

| From Protection Scenario | To Protection Scenario[1] | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| 0 | X | X | X | X | | | | |
| 1 | | X | | X | | | | |
| 2 | | | X | X | | | | |
| 3 | | | | X | | | | |
| 4 | | | | X | X | | | |
| 5 | | | X | X | X | X | | |
| 6 | | X | | X | X | | X | |
| 7 | X | X | X | X | X | X | X | X |

1. Allowed transitions marked with X, see Figure 24-14 for a definition of the scenarios.

### 24.3.2.10 EEE Protection Register (EPROT)

The EPROT register defines which buffer RAM EEE partition areas are protected against writes.

Offset Module Base + 0x0009

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | EPOPEN | RNV[6:4] | | | EPDIS | EPS[2:0] | | |
| W | | | | | | | | |
| Reset | F | F | F | F | F | F | F | F |

= Unimplemented or Reserved

**Figure 24-15. EEE Protection Register (EPROT)**

All bits in the EPROT register are readable and writable except for RNV[6:4] which are only readable. The EPOPEN and EPDIS bits can only be written to the protected state. The EPS bits can be written anytime until the EPDIS bit is cleared. If the EPOPEN bit is cleared, the state of the EPDIS and EPS bits is irrelevant.

During the reset sequence, the EPROT register is loaded from the EEE protection byte in the Flash configuration field at global address 0x7F_FF0D located in P-Flash memory (see Table 24-3) as indicated by reset condition F in Figure 24-15. To change the EEE protection that will be loaded during the reset sequence, the P-Flash sector containing the EEE protection byte must be unprotected, then the EEE protection byte must be programmed. If a double bit fault is detected while reading the P-Flash phrase

containing the EEE protection byte during the reset sequence, the EPOPEN bit will be cleared and remaining bits in the EPROT register will be set to leave the buffer RAM EEE partition fully protected.

Trying to write data to any protected area in the buffer RAM EEE partition will result in a protection violation error and the EPVIOLIF flag will be set in the FERSTAT register. Trying to write data to any protected area in the buffer RAM partitioned for user access will not be prevented and the EPVIOLIF flag in the FERSTAT register will not set.

**Table 24-24. EPROT Field Descriptions**

| Field | Description |
|---|---|
| 7<br>EPOPEN | **Enables writes to the Buffer RAM partitioned for EEE**<br>0 The entire buffer RAM EEE partition is protected from writes<br>1 Unprotected buffer RAM EEE partition areas are enabled for writes |
| 6–4<br>RNV[6:4] | **Reserved Nonvolatile Bits** — The RNV bits should remain in the erased state for future enhancements |
| 3<br>EPDIS | **Buffer RAM Protection Address Range Disable** — The EPDIS bit determines whether there is a protected area in a specific region of the buffer RAM EEE partition.<br>0 Protection enabled<br>1 Protection disabled |
| 2–0<br>EPS[2:0] | **Buffer RAM Protection Size** — The EPS[2:0] bits determine the size of the protected area in the buffer RAM EEE partition as shown inTable 24-21. The EPS bits can only be written to while the EPDIS bit is set. |

**Table 24-25. Buffer RAM EEE Partition Protection Address Range**

| EPS[2:0] | Global Address Range | Protected Size |
|---|---|---|
| 000 | 0x13_FFC0 – 0x13_FFFF | 64 bytes |
| 001 | 0x13_FF80 – 0x13_FFFF | 128 bytes |
| 010 | 0x13_FF40 – 0x13_FFFF | 192 bytes |
| 011 | 0x13_FF00 – 0x13_FFFF | 256 bytes |
| 100 | 0x13_FEC0 – 0x13_FFFF | 320 bytes |
| 101 | 0x13_FE80 – 0x13_FFFF | 384 bytes |
| 110 | 0x13_FE40 – 0x13_FFFF | 448 bytes |
| 111 | 0x13_FE00 – 0x13_FFFF | 512 bytes |

### 24.3.2.11 Flash Common Command Object Register (FCCOB)

The FCCOB is an array of six words addressed via the CCOBIX index found in the FCCOBIX register. Byte wide reads and writes are allowed to the FCCOB register.

Offset Module Base + 0x000A

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | CCOB[15:8] | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 24-16. Flash Common Command Object High Register (FCCOBHI)**

Offset Module Base + 0x000B

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | CCOB[7:0] | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 24-17. Flash Common Command Object Low Register (FCCOBLO)**

### 24.3.2.11.1 FCCOB - NVM Command Mode

NVM command mode uses the indexed FCCOB register to provide a command code and its relevant parameters to the Memory Controller. The user first sets up all required FCCOB fields and then initiates the command's execution by writing a 1 to the CCIF bit in the FSTAT register (a 1 written by the user clears the CCIF command completion flag to 0). When the user clears the CCIF bit in the FSTAT register all FCCOB parameter fields are locked and cannot be changed by the user until the command completes (as evidenced by the Memory Controller returning CCIF to 1). Some commands return information to the FCCOB register array.

The generic format for the FCCOB parameter fields in NVM command mode is shown in Table 24-26. The return values are available for reading after the CCIF flag in the FSTAT register has been returned to 1 by the Memory Controller. Writes to the unimplemented parameter fields (CCOBIX = 110 and CCOBIX = 111) are ignored with reads from these fields returning 0x0000.

Table 24-26 shows the generic Flash command format. The high byte of the first word in the CCOB array contains the command code, followed by the parameters for this specific Flash command. For details on the FCCOB settings required by each command, see the Flash command descriptions in Section 24.4.2.

**Table 24-26. FCCOB - NVM Command Mode (Typical Usage)**

| CCOBIX[2:0] | Byte | FCCOB Parameter Fields (NVM Command Mode) |
|---|---|---|
| 000 | HI | FCMD[7:0] defining Flash command |
| | LO | 0, Global address [22:16] |
| 001 | HI | Global address [15:8] |
| | LO | Global address [7:0] |
| 010 | HI | Data 0 [15:8] |
| | LO | Data 0 [7:0] |

**Table 24-26. FCCOB - NVM Command Mode (Typical Usage)**

| CCOBIX[2:0] | Byte | FCCOB Parameter Fields (NVM Command Mode) |
|---|---|---|
| 011 | HI | Data 1 [15:8] |
| | LO | Data 1 [7:0] |
| 100 | HI | Data 2 [15:8] |
| | LO | Data 2 [7:0] |
| 101 | HI | Data 3 [15:8] |
| | LO | Data 3 [7:0] |

### 24.3.2.12 EEE Tag Counter Register (ETAG)

The ETAG register contains the number of outstanding words in the buffer RAM EEE partition that need to be programmed into the D-Flash EEE partition. The ETAG register is decremented prior to the related tagged word being programmed into the D-Flash EEE partition. All tagged words have been programmed into the D-Flash EEE partition once all bits in the ETAG register read 0 and the MGBUSY flag in the FSTAT register reads 0.

Offset Module Base + 0x000C

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | ETAG[15:8] | | | | | | | |
| W | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented or Reserved

**Figure 24-18. EEE Tag Counter High Register (ETAGHI)**

Offset Module Base + 0x000D

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | ETAG[7:0] | | | | | | | |
| W | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented or Reserved

**Figure 24-19. EEE Tag Counter Low Register (ETAGLO)**

All ETAG bits are readable but not writable and are cleared by the Memory Controller.

### 24.3.2.13 Flash ECC Error Results Register (FECCR)

The FECCR registers contain the result of a detected ECC fault for both single bit and double bit faults. The FECCR register provides access to several ECC related fields as defined by the ECCRIX index bits in the FECCRIX register (see Section 24.3.2.4). Once ECC fault information has been stored, no other

fault information will be recorded until the specific ECC fault flag has been cleared. In the event of simultaneous ECC faults, the priority for fault recording is:

1. Double bit fault over single bit fault
2. CPU over XGATE

Offset Module Base + 0x000E

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | ECCR[15:8] | | | | | | | |
| W | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented or Reserved

**Figure 24-20. Flash ECC Error Results High Register (FECCRHI)**

Offset Module Base + 0x000F

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | ECCR[7:0] | | | | | | | |
| W | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented or Reserved

**Figure 24-21. Flash ECC Error Results Low Register (FECCRLO)**

All FECCR bits are readable but not writable.

**Table 24-27. FECCR Index Settings**

| ECCRIX[2:0] | FECCR Register Content | | |
|---|---|---|---|
| | **Bits [15:8]** | **Bit[7]** | **Bits[6:0]** |
| 000 | Parity bits read from Flash block | CPU or XGATE source identity | Global address [22:16] |
| 001 | Global address [15:0] | | |
| 010 | Data 0 [15:0] | | |
| 011 | Data 1 [15:0] (P-Flash only) | | |
| 100 | Data 2 [15:0] (P-Flash only) | | |
| 101 | Data 3 [15:0] (P-Flash only) | | |
| 110 | Not used, returns 0x0000 when read | | |
| 111 | Not used, returns 0x0000 when read | | |

**Table 24-28. FECCR Index=000 Bit Descriptions**

| Field | Description |
|---|---|
| 15:8<br>PAR[7:0] | **ECC Parity Bits** — Contains the 8 parity bits from the 72 bit wide P-Flash data word or the 6 parity bits, allocated to PAR[5:0], from the 22 bit wide D-Flash word with PAR[7:6]=00. |
| 7<br>XBUS01 | **Bus Source Identifier** — The XBUS01 bit determines whether the ECC error was caused by a read access from the CPU or XGATE.<br>0 ECC Error happened on the CPU access<br>1 ECC Error happened on the XGATE access |
| 6–0<br>GADDR[22:16] | **Global Address** — The GADDR[22:16] field contains the upper seven bits of the global address having caused the error. |

The P-Flash word addressed by ECCRIX = 001 contains the lower 16 bits of the global address. The following four words addressed by ECCRIX = 010 to 101 contain the 64-bit wide data phrase. The four data words and the parity byte are the uncorrected data read from the P-Flash block.

The D-Flash word addressed by ECCRIX = 001 contains the lower 16 bits of the global address. The uncorrected 16-bit data word is addressed by ECCRIX = 010.

### 24.3.2.14 Flash Option Register (FOPT)

The FOPT register is the Flash option register.

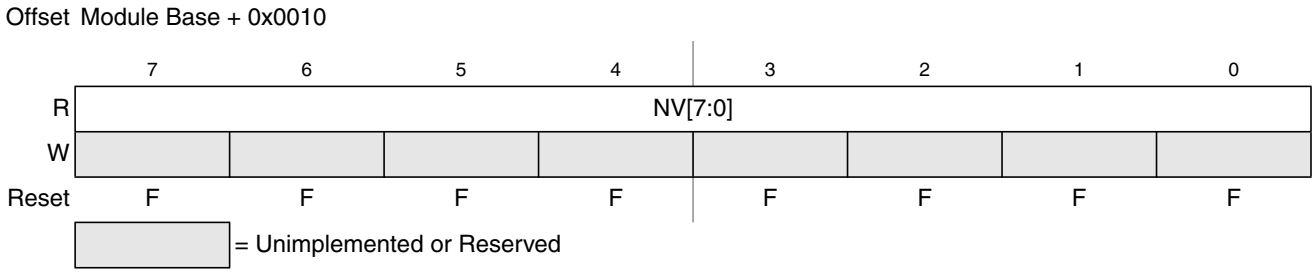



**Figure 24-22. Flash Option Register (FOPT)**

All bits in the FOPT register are readable but are not writable.

During the reset sequence, the FOPT register is loaded from the Flash nonvolatile byte in the Flash configuration field at global address 0x7F_FF0E located in P-Flash memory (see Table 24-3) as indicated by reset condition F in Figure 24-22. If a double bit fault is detected while reading the P-Flash phrase containing the Flash nonvolatile byte during the reset sequence, all bits in the FOPT register will be set.

**Table 24-29. FOPT Field Descriptions**

| Field | Description |
|---|---|
| 7–0<br>NV[7:0] | **Nonvolatile Bits** — The NV[7:0] bits are available as nonvolatile bits. Refer to the device user guide for proper use of the NV bits. |

### 24.3.2.15 Flash Reserved0 Register (FRSV0)

This Flash register is reserved for factory testing.

Offset Module Base + 0x0011

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| W | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented or Reserved

**Figure 24-23. Flash Reserved0 Register (FRSV0)**

All bits in the FRSV0 register read 0 and are not writable.

### 24.3.2.16 Flash Reserved1 Register (FRSV1)

This Flash register is reserved for factory testing.

Offset Module Base + 0x0012

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| W | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented or Reserved

**Figure 24-24. Flash Reserved1 Register (FRSV1)**

All bits in the FRSV1 register read 0 and are not writable.

### 24.3.2.17 Flash Reserved2 Register (FRSV2)

This Flash register is reserved for factory testing.

Offset Module Base + 0x0013

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| W | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented or Reserved

**Figure 24-25. Flash Reserved2 Register (FRSV2)**

All bits in the FRSV2 register read 0 and are not writable.

## 24.4 Functional Description

### 24.4.1 Flash Command Operations

Flash command operations are used to modify Flash memory contents or configure module resources for EEE operation.

The next sections describe:

- How to write the FCLKDIV register that is used to generate a time base (FCLK) derived from OSCCLK for Flash program and erase command operations
- The command write sequence used to set Flash command parameters and launch execution
- Valid Flash commands available for execution

#### 24.4.1.1 Writing the FCLKDIV Register

Prior to issuing any Flash program or erase command after a reset, the user is required to write the FCLKDIV register to divide OSCCLK down to a target FCLK of 1 MHz. Table 24-9 shows recommended values for the FDIV field based on OSCCLK frequency.

**NOTE**

Programming or erasing the Flash memory cannot be performed if the bus clock runs at less than 1 MHz. Setting FDIV too high can destroy the Flash memory due to overstress. Setting FDIV too low can result in incomplete programming or erasure of the Flash memory cells.

When the FCLKDIV register is written, the FDIVLD bit is set automatically. If the FDIVLD bit is 0, the FCLKDIV register has not been written since the last reset. If the FCLKDIV register has not been written, any Flash program or erase command loaded during a command write sequence will not execute and the ACCERR bit in the FSTAT register will set.

#### 24.4.1.2 Command Write Sequence

The Memory Controller will launch all valid Flash commands entered using a command write sequence.

Before launching a command, the ACCERR and FPVIOL bits in the FSTAT register must be clear (see Section 24.3.2.7) and the CCIF flag should be tested to determine the status of the current command write sequence. If CCIF is 0, the previous command write sequence is still active, a new command write sequence cannot be started, and all writes to the FCCOB register are ignored.

**CAUTION**

Writes to any Flash register must be avoided while a Flash command is active (CCIF=0) to prevent corruption of Flash register contents and Memory Controller behavior.

#### 24.4.1.2.1 Define FCCOB Contents

The FCCOB parameter fields must be loaded with all required parameters for the Flash command being executed. Access to the FCCOB parameter fields is controlled via the CCOBIX bits in the FCCOBIX register (see Section 24.3.2.3).

The contents of the FCCOB parameter fields are transferred to the Memory Controller when the user clears the CCIF command completion flag in the FSTAT register (writing 1 clears the CCIF to 0). The CCIF flag will remain clear until the Flash command has completed. Upon completion, the Memory Controller will return CCIF to 1 and the FCCOB register will be used to communicate any results. The flow for a generic command write sequence is shown in Figure 24-26.

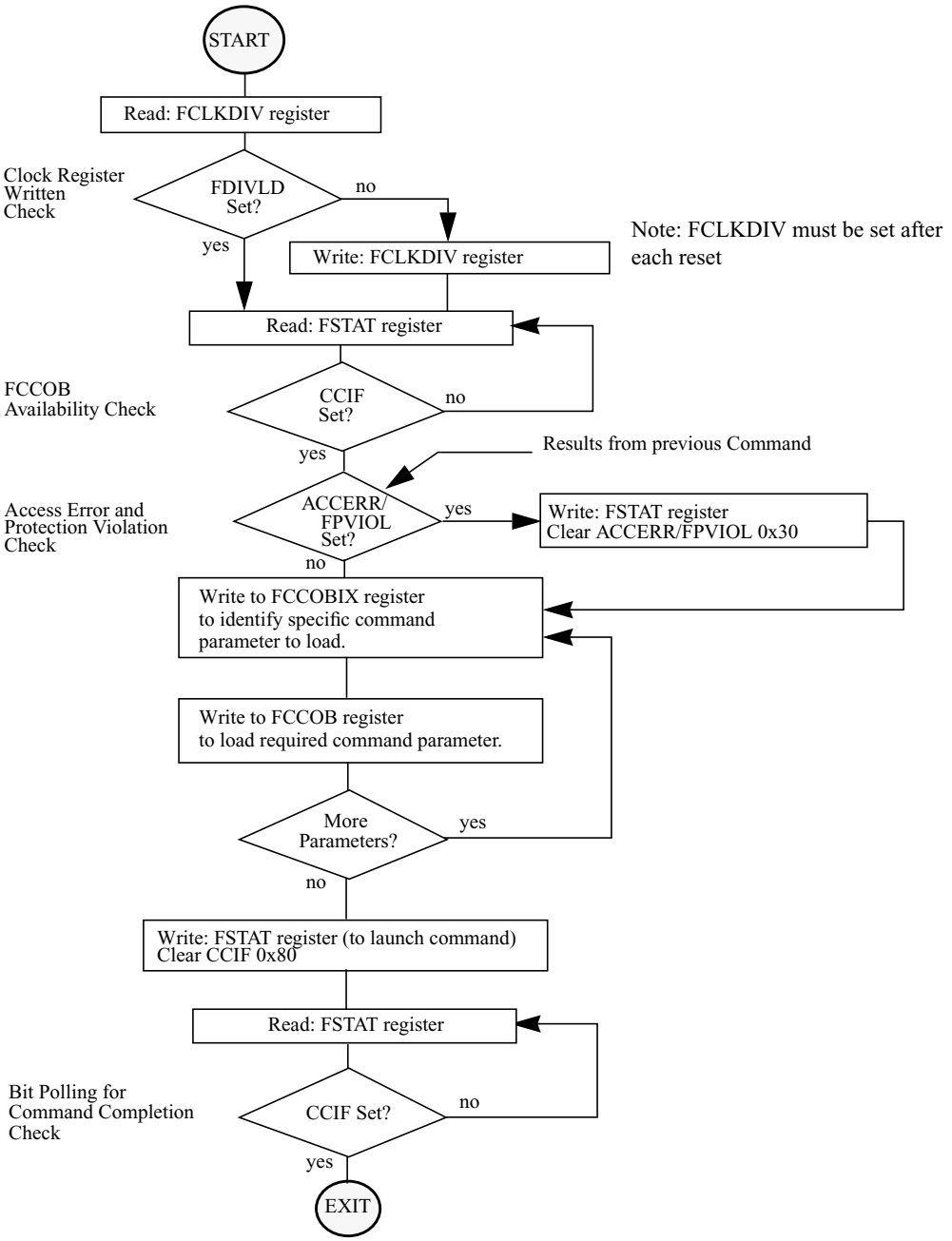



**Figure 24-26. Generic Flash Command Write Sequence Flowchart**

### 24.4.1.3 Valid Flash Module Commands

**Table 24-30. Flash Commands by Mode**

| FCMD | Command | Unsecured | | | | Secured | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | NS (1) | NX (2) | SS(3) | ST(4) | NS (5) | NX (6) | SS(7) | ST(8) |
| 0x01 | Erase Verify All Blocks | * | * | * | * | * | * | * | * |
| 0x02 | Erase Verify Block | * | * | * | * | * | * | * | * |
| 0x03 | Erase Verify P-Flash Section | * | * | * | * | * | | | |
| 0x04 | Read Once | * | * | * | * | * | | | |
| 0x05 | Reserved | * | * | * | * | * | | | |
| 0x06 | Program P-Flash | * | * | * | * | * | | | |
| 0x07 | Program Once | * | * | * | * | * | | | |
| 0x08 | Erase All Blocks | | | * | * | | | * | * |
| 0x09 | Erase P-Flash Block | * | * | * | * | * | | | |
| 0x0A | Erase P-Flash Sector | * | * | * | * | * | | | |
| 0x0B | Unsecure Flash | | | * | * | | | * | * |
| 0x0C | Verify Backdoor Access Key | * | | | | * | | | |
| 0x0D | Set User Margin Level | * | * | * | * | * | | | |
| 0x0E | Set Field Margin Level | | | * | * | | | | |
| 0x0F | Full Partition D-Flash | | | * | * | | | | |
| 0x10 | Erase Verify D-Flash Section | * | * | * | * | * | | | |
| 0x11 | Program D-Flash | * | * | * | * | * | | | |
| 0x12 | Erase D-Flash Sector | * | * | * | * | * | | | |
| 0x13 | Enable EEPROM Emulation | * | * | * | * | * | * | * | * |
| 0x14 | Disable EEPROM Emulation | * | * | * | * | * | * | * | * |
| 0x15 | EEPROM Emulation Query | * | * | * | * | * | * | * | * |
| 0x20 | Partition D-Flash | * | * | * | * | * | * | * | * |

1. Unsecured Normal Single Chip mode.
2. Unsecured Normal Expanded mode.
3. Unsecured Special Single Chip mode.
4. Unsecured Special Mode.
5. Secured Normal Single Chip mode.
6. Secured Normal Expanded mode.
7. Secured Special Single Chip mode.
8. Secured Special Mode.

### 24.4.1.4 P-Flash Commands

Table 24-31 summarizes the valid P-Flash commands along with the effects of the commands on the P-Flash block and other resources within the Flash module.

**Table 24-31. P-Flash Commands**

| FCMD | Command | Function on P-Flash Memory |
|---|---|---|
| 0x01 | Erase Verify All Blocks | Verify that all P-Flash (and D-Flash) blocks are erased. |
| 0x02 | Erase Verify Block | Verify that a P-Flash block is erased. |
| 0x03 | Erase Verify P-Flash Section | Verify that a given number of words starting at the address provided are erased. |
| 0x04 | Read Once | Read a dedicated 64 byte field in the nonvolatile information register in P-Flash block 0 that was previously programmed using the Program Once command. |
| 0x06 | Program P-Flash | Program a phrase in a P-Flash block. |
| 0x07 | Program Once | Program a dedicated 64 byte field in the nonvolatile information register in P-Flash block 0 that is allowed to be programmed only once. |
| 0x08 | Erase All Blocks | Erase all P-Flash (and D-Flash) blocks.<br>An erase of all Flash blocks is only possible when the FPLDIS, FPHDIS, and FPOPEN bits in the FPROT register and the EPDIS and EPOPEN bits in the EPROT register are set prior to launching the command. |
| 0x09 | Erase P-Flash Block | Erase a single P-Flash block.<br>An erase of the full P-Flash block is only possible when FPLDIS, FPHDIS and FPOPEN bits in the FPROT register are set prior to launching the command. |
| 0x0A | Erase P-Flash Sector | Erase all bytes in a P-Flash sector. |
| 0x0B | Unsecure Flash | Supports a method of releasing MCU security by erasing all P-Flash (and D-Flash) blocks and verifying that all P-Flash (and D-Flash) blocks are erased. |
| 0x0C | Verify Backdoor Access Key | Supports a method of releasing MCU security by verifying a set of security keys. |
| 0x0D | Set User Margin Level | Specifies a user margin read level for all P-Flash blocks. |
| 0x0E | Set Field Margin Level | Specifies a field margin read level for all P-Flash blocks (special modes only). |

### 24.4.1.5 D-Flash and EEE Commands

Table 24-32 summarizes the valid D-Flash and EEE commands along with the effects of the commands on the D-Flash block and EEE operation.

**Table 24-32. D-Flash Commands**

| FCMD | Command | Function on D-Flash Memory |
|---|---|---|
| 0x01 | Erase Verify All Blocks | Verify that all D-Flash (and P-Flash) blocks are erased. |
| 0x02 | Erase Verify Block | Verify that the D-Flash block is erased. |
| 0x08 | Erase All Blocks | Erase all D-Flash (and P-Flash) blocks.<br>An erase of all Flash blocks is only possible when the FPLDIS, FPHDIS, and FPOPEN bits in the FPROT register and the EPDIS and EPOPEN bits in the EPROT register are set prior to launching the command. |

**Table 24-32. D-Flash Commands**

| FCMD | Command | Function on D-Flash Memory |
|---|---|---|
| 0x0B | Unsecure Flash | Supports a method of releasing MCU security by erasing all D-Flash (and P-Flash) blocks and verifying that all D-Flash (and P-Flash) blocks are erased. |
| 0x0D | Set User Margin Level | Specifies a user margin read level for the D-Flash block. |
| 0x0E | Set Field Margin Level | Specifies a field margin read level for the D-Flash block (special modes only). |
| 0x0F | Full Partition D-Flash | Erase the D-Flash block and partition an area of the D-Flash block for user access. |
| 0x10 | Erase Verify D-Flash Section | Verify that a given number of words starting at the address provided are erased. |
| 0x11 | Program D-Flash | Program up to four words in the D-Flash block. |
| 0x12 | Erase D-Flash Sector | Erase all bytes in a sector of the D-Flash block. |
| 0x13 | Enable EEPROM Emulation | Enable EEPROM emulation where writes to the buffer RAM EEE partition will be copied to the D-Flash EEE partition. |
| 0x14 | Disable EEPROM Emulation | Suspend all current erase and program activity related to EEPROM emulation but leave current EEE tags set. |
| 0x15 | EEPROM Emulation Query | Returns EEE partition and status variables. |
| 0x20 | Partition D-Flash | Partition an area of the D-Flash block for user access. |

## 24.4.2 Flash Command Description

This section provides details of all available Flash commands launched by a command write sequence. The ACCERR bit in the FSTAT register will be set during the command write sequence if any of the following illegal steps are performed, causing the command not to be processed by the Memory Controller:

- Starting any command write sequence that programs or erases Flash memory before initializing the FCLKDIV register
- Writing an invalid command as part of the command write sequence
- For additional possible errors, refer to the error handling table provided for each command

If a Flash block is read during execution of an algorithm (CCIF = 0) on that same block, the read operation will return invalid data. If the SFDIF or DFDIF flags were not previously set when the invalid read operation occurred, both the SFDIF and DFDIF flags will be set and the FECCR registers will be loaded with the global address used in the invalid read operation with the data and parity fields set to all 0.

If the ACCERR or FPVIOL bits are set in the FSTAT register, the user must clear these bits before starting any command write sequence (see Section 24.3.2.7).

**CAUTION**

A Flash word or phrase must be in the erased state before being programmed. Cumulative programming of bits within a Flash word or phrase is not allowed.

### 24.4.2.1 Erase Verify All Blocks Command

The Erase Verify All Blocks command will verify that all P-Flash and D-Flash blocks have been erased.

**Table 24-33. Erase Verify All Blocks Command FCCOB Requirements**

| CCOBIX[2:0] | FCCOB Parameters | |
|---|---|---|
| 000 | 0x01 | Not required |

Upon clearing CCIF to launch the Erase Verify All Blocks command, the Memory Controller will verify that the entire Flash memory space is erased. The CCIF flag will set after the Erase Verify All Blocks operation has completed.

**Table 24-34. Erase Verify All Blocks Command Error Handling**

| Register | Error Bit | Error Condition |
|---|---|---|
| FSTAT | ACCERR | Set if CCOBIX[2:0] != 000 at command launch |
| | FPVIOL | None |
| | MGSTAT1 | Set if any errors have been encountered during the read(1) |
| | MGSTAT0 | Set if any non-correctable errors have been encountered during the read[1] |
| FERSTAT | EPVIOLIF | None |

1. As found in the memory map for FTM256K2.

### 24.4.2.2 Erase Verify Block Command

The Erase Verify Block command allows the user to verify that an entire P-Flash or D-Flash block has been erased. The FCCOB upper global address bits determine which block must be verified.

**Table 24-35. Erase Verify Block Command FCCOB Requirements**

| CCOBIX[2:0] | FCCOB Parameters | |
|---|---|---|
| 000 | 0x02 | Global address [22:16] of the Flash block to be verified. |

Upon clearing CCIF to launch the Erase Verify Block command, the Memory Controller will verify that the selected P-Flash or D-Flash block is erased. The CCIF flag will set after the Erase Verify Block operation has completed.

**Table 24-36. Erase Verify Block Command Error Handling**

| Register | Error Bit | Error Condition |
|---|---|---|
| FSTAT | ACCERR | Set if CCOBIX[2:0] != 000 at command launch |
| | | Set if an invalid global address [22:16] is supplied(1) |
| | FPVIOL | None |
| | MGSTAT1 | Set if any errors have been encountered during the read(2) |
| | MGSTAT0 | Set if any non-correctable errors have been encountered during the read2 |
| FERSTAT | EPVIOLIF | None |

1. As defined by the memory map for FTM256K2.
2. As found in the memory map for FTM256K2.

### 24.4.2.3 Erase Verify P-Flash Section Command

The Erase Verify P-Flash Section command will verify that a section of code in the P-Flash memory is erased. The Erase Verify P-Flash Section command defines the starting point of the code to be verified and the number of phrases.128

**Table 24-37. Erase Verify P-Flash Section Command FCCOB Requirements**

| CCOBIX[2:0] | FCCOB Parameters | |
|---|---|---|
| 000 | 0x03 | Global address [22:16] of a P-Flash block |
| 001 | Global address [15:0] of the first phrase to be verified | |
| 010 | Number of phrases to be verified | |

Upon clearing CCIF to launch the Erase Verify P-Flash Section command, the Memory Controller will verify the selected section of Flash memory is erased. The CCIF flag will set after the Erase Verify P-Flash Section operation has completed.

**Table 24-38. Erase Verify P-Flash Section Command Error Handling**

| Register | Error Bit | Error Condition |
|---|---|---|
| FSTAT | ACCERR | Set if CCOBIX[2:0] != 010 at command launch |
| | | Set if command not available in current mode (see Table 24-30) |
| | | Set if an invalid global address [22:0] is supplied(1) |
| | | Set if a misaligned phrase address is supplied (global address [2:0] != 000) |
| | | Set if the requested section crosses a 128 Kbyte boundary |
| | FPVIOL | None |
| | MGSTAT1 | Set if any errors have been encountered during the read(2) |
| | MGSTAT0 | Set if any non-correctable errors have been encountered during the read2 |
| FERSTAT | EPVIOLIF | None |

1. As defined by the memory map for FTM256K2.
2. As found in the memory map for FTM256K2.

### 24.4.2.4 Read Once Command

The Read Once command provides read access to a reserved 64 byte field (8 phrases) located in the nonvolatile information register of P-Flash block 0. The Read Once field is programmed using the Program Once command described in Section 24.4.2.6. The Read Once command must not be executed from the Flash block containing the Program Once reserved field to avoid code runaway.

**Table 24-39. Read Once Command FCCOB Requirements**

| CCOBIX[2:0] | FCCOB Parameters | |
|---|---|---|
| 000 | 0x04 | Not Required |
| 001 | Read Once phrase index (0x0000 - 0x0007) | |
| 010 | Read Once word 0 value | |
| 011 | Read Once word 1 value | |
| 100 | Read Once word 2 value | |
| 101 | Read Once word 3 value | |

Upon clearing CCIF to launch the Read Once command, a Read Once phrase is fetched and stored in the FCCOB indexed register. The CCIF flag will set after the Read Once operation has completed. Valid phrase index values for the Read Once command range from 0x0000 to 0x0007. During execution of the Read Once command, any attempt to read addresses within P-Flash block 0 will return invalid data.

**Table 24-40. Read Once Command Error Handling**

| Register | Error Bit | Error Condition |
|---|---|---|
| FSTAT | ACCERR | Set if CCOBIX[2:0] != 001 at command launch |
| | | Set if command not available in current mode (see Table 24-30) |
| | | Set if an invalid phrase index is supplied |
| | FPVIOL | None |
| | MGSTAT1 | Set if any errors have been encountered during the read |
| | MGSTAT0 | Set if any non-correctable errors have been encountered during the read |
| FERSTAT | EPVIOLIF | None |

### 24.4.2.5 Program P-Flash Command

The Program P-Flash operation will program a previously erased phrase in the P-Flash memory using an embedded algorithm.

**CAUTION**

A P-Flash phrase must be in the erased state before being programmed. Cumulative programming of bits within a Flash phrase is not allowed.

**Table 24-41. Program P-Flash Command FCCOB Requirements**

| CCOBIX[2:0] | FCCOB Parameters | |
|---|---|---|
| 000 | 0x06 | Global address [22:16] to identify P-Flash block |
| 001 | Global address [15:0] of phrase location to be programmed(1) | |
| 010 | Word 0 program value | |
| 011 | Word 1 program value | |
| 100 | Word 2 program value | |
| 101 | Word 3 program value | |

1. Global address [2:0] must be 000

Upon clearing CCIF to launch the Program P-Flash command, the Memory Controller will program the data words to the supplied global address and will then proceed to verify the data words read back as expected. The CCIF flag will set after the Program P-Flash operation has completed.

**Table 24-42. Program P-Flash Command Error Handling**

| Register | Error Bit | Error Condition |
|---|---|---|
| FSTAT | ACCERR | Set if CCOBIX[2:0] != 101 at command launch |
| | | Set if command not available in current mode (see Table 24-30) |
| | | Set if an invalid global address [22:0] is supplied(1) |
| | | Set if a misaligned phrase address is supplied (global address [2:0] != 000) |
| | FPVIOL | Set if the global address [22:0] points to a protected area |
| | MGSTAT1 | Set if any errors have been encountered during the verify operation(2) |
| | MGSTAT0 | Set if any non-correctable errors have been encountered during the verify operation2 |
| FERSTAT | EPVIOLIF | None |

1. As defined by the memory map for FTM256K2.
2. As found in the memory map for FTM256K2.

### 24.4.2.6 Program Once Command

The Program Once command restricts programming to a reserved 64 byte field (8 phrases) in the nonvolatile information register located in P-Flash block 0. The Program Once reserved field can be read using the Read Once command as described in Section 24.4.2.4. The Program Once command must only be issued once since the nonvolatile information register in P-Flash block 0 cannot be erased. The Program

Once command must not be executed from the Flash block containing the Program Once reserved field to avoid code runaway.

**Table 24-43. Program Once Command FCCOB Requirements**

| CCOBIX[2:0] | FCCOB Parameters | |
|---|---|---|
| 000 | 0x07 | Not Required |
| 001 | Program Once phrase index (0x0000 - 0x0007) | |
| 010 | Program Once word 0 value | |
| 011 | Program Once word 1 value | |
| 100 | Program Once word 2 value | |
| 101 | Program Once word 3 value | |

Upon clearing CCIF to launch the Program Once command, the Memory Controller first verifies that the selected phrase is erased. If erased, then the selected phrase will be programmed and then verified with read back. The CCIF flag will remain clear, setting only after the Program Once operation has completed.

The reserved nonvolatile information register accessed by the Program Once command cannot be erased and any attempt to program one of these phrases a second time will not be allowed. Valid phrase index values for the Program Once command range from 0x0000 to 0x0007. During execution of the Program Once command, any attempt to read addresses within P-Flash block 0 will return invalid data.

**Table 24-44. Program Once Command Error Handling**

| Register | Error Bit | Error Condition |
|---|---|---|
| FSTAT | ACCERR | Set if CCOBIX[2:0] != 101 at command launch |
| | | Set if command not available in current mode (see Table 24-30) |
| | | Set if an invalid phrase index is supplied |
| | | Set if the requested phrase has already been programmed(1) |
| | FPVIOL | None |
| | MGSTAT1 | Set if any errors have been encountered during the verify operation |
| | MGSTAT0 | Set if any non-correctable errors have been encountered during the verify operation |
| FERSTAT | EPVIOLIF | None |

1. If a Program Once phrase is initially programmed to 0xFFFF_FFFF_FFFF_FFFF, the Program Once command will be allowed to execute again on that same phrase.

### 24.4.2.7 Erase All Blocks Command

The Erase All Blocks operation will erase the entire P-Flash and D-Flash memory space including the EEE nonvolatile information register.

**Table 24-45. Erase All Blocks Command FCCOB Requirements**

| CCOBIX[2:0] | FCCOB Parameters | |
|---|---|---|
| 000 | 0x08 | Not required |

Upon clearing CCIF to launch the Erase All Blocks command, the Memory Controller will erase the entire Flash memory space and verify that it is erased. If the Memory Controller verifies that the entire Flash memory space was properly erased, security will be released. During the execution of this command (CCIF=0) the user must not write to any Flash module register. The CCIF flag will set after the Erase All Blocks operation has completed.

**Table 24-46. Erase All Blocks Command Error Handling**

| Register | Error Bit | Error Condition |
|---|---|---|
| FSTAT | ACCERR | Set if CCOBIX[2:0] != 000 at command launch |
| | | Set if command not available in current mode (see Table 24-30) |
| | FPVIOL | Set if any area of the P-Flash memory is protected |
| | MGSTAT1 | Set if any errors have been encountered during the verify operation(1) |
| | MGSTAT0 | Set if any non-correctable errors have been encountered during the verify operation[1] |
| FERSTAT | EPVIOLIF | Set if any area of the buffer RAM EEE partition is protected |

1. As found in the memory map for FTM256K2.

### 24.4.2.8 Erase P-Flash Block Command

The Erase P-Flash Block operation will erase all addresses in a P-Flash block.

**Table 24-47. Erase P-Flash Block Command FCCOB Requirements**

| CCOBIX[2:0] | FCCOB Parameters | |
|---|---|---|
| 000 | 0x09 | Global address [22:16] to identify P-Flash block |
| 001 | Global address [15:0] in P-Flash block to be erased | |

Upon clearing CCIF to launch the Erase P-Flash Block command, the Memory Controller will erase the selected P-Flash block and verify that it is erased. The CCIF flag will set after the Erase P-Flash Block operation has completed.

**Table 24-48. Erase P-Flash Block Command Error Handling**

| Register | Error Bit | Error Condition |
|---|---|---|
| FSTAT | ACCERR | Set if CCOBIX[2:0] != 001 at command launch |
| | | Set if command not available in current mode (see Table 24-30) |
| | | Set if an invalid global address [22:16] is supplied(1) |
| | FPVIOL | Set if an area of the selected P-Flash block is protected |
| | MGSTAT1 | Set if any errors have been encountered during the verify operation(2) |
| | MGSTAT0 | Set if any non-correctable errors have been encountered during the verify operation2 |
| FERSTAT | EPVIOLIF | None |

1. As defined by the memory map for FTM256K2.
2. As found in the memory map for FTM256K2.

### 24.4.2.9 Erase P-Flash Sector Command

The Erase P-Flash Sector operation will erase all addresses in a P-Flash sector.

**Table 24-49. Erase P-Flash Sector Command FCCOB Requirements**

| CCOBIX[2:0] | FCCOB Parameters | |
|---|---|---|
| 000 | 0x0A | Global address [22:16] to identify P-Flash block to be erased |
| 001 | Global address [15:0] anywhere within the sector to be erased. Refer to Section 24.1.2.1 for the P-Flash sector size. | |

Upon clearing CCIF to launch the Erase P-Flash Sector command, the Memory Controller will erase the selected Flash sector and then verify that it is erased. The CCIF flag will be set after the Erase P-Flash Sector operation has completed.

**Table 24-50. Erase P-Flash Sector Command Error Handling**

| Register | Error Bit | Error Condition |
|---|---|---|
| FSTAT | ACCERR | Set if CCOBIX[2:0] != 001 at command launch |
| | | Set if command not available in current mode (see Table 24-30) |
| | | Set if an invalid global address [22:16] is supplied(1) |
| | | Set if a misaligned phrase address is supplied (global address [2:0] != 000) |
| | FPVIOL | Set if the selected P-Flash sector is protected |
| | MGSTAT1 | Set if any errors have been encountered during the verify operation |
| | MGSTAT0 | Set if any non-correctable errors have been encountered during the verify operation |
| FERSTAT | EPVIOLIF | None |

1. As defined by the memory map for FTM256K2.

### 24.4.2.10 Unsecure Flash Command

The Unsecure Flash command will erase the entire P-Flash and D-Flash memory space and, if the erase is successful, will release security.

**Table 24-51. Unsecure Flash Command FCCOB Requirements**

| CCOBIX[2:0] | FCCOB Parameters | |
|---|---|---|
| 000 | 0x0B | Not required |

Upon clearing CCIF to launch the Unsecure Flash command, the Memory Controller will erase the entire P-Flash and D-Flash memory space and verify that it is erased. If the Memory Controller verifies that the entire Flash memory space was properly erased, security will be released. If the erase verify is not successful, the Unsecure Flash operation sets MGSTAT1 and terminates without changing the security state. During the execution of this command (CCIF=0) the user must not write to any Flash module register. The CCIF flag is set after the Unsecure Flash operation has completed.

**Table 24-52. Unsecure Flash Command Error Handling**

| Register | Error Bit | Error Condition |
|---|---|---|
| FSTAT | ACCERR | Set if CCOBIX[2:0] != 000 at command launch |
| | | Set if command not available in current mode (see Table 24-30) |
| | FPVIOL | Set if any area of the P-Flash memory is protected |
| | MGSTAT1 | Set if any errors have been encountered during the verify operation(1) |
| | MGSTAT0 | Set if any non-correctable errors have been encountered during the verify operation[1] |
| FERSTAT | EPVIOLIF | Set if any area of the buffer RAM EEE partition is protected |

1. As found in the memory map for FTM256K2.

### 24.4.2.11 Verify Backdoor Access Key Command

The Verify Backdoor Access Key command will only execute if it is enabled by the KEYEN bits in the FSEC register (see Table 24-11). The Verify Backdoor Access Key command releases security if user-supplied keys match those stored in the Flash security bytes of the Flash configuration field (see Table 24-3). The Verify Backdoor Access Key command must not be executed from the Flash block containing the backdoor comparison key to avoid code runaway.

**Table 24-53. Verify Backdoor Access Key Command FCCOB Requirements**

| CCOBIX[2:0] | FCCOB Parameters | |
|---|---|---|
| 000 | 0x0C | Not required |
| 001 | Key 0 | |
| 010 | Key 1 | |

**Table 24-53. Verify Backdoor Access Key Command FCCOB Requirements**

| CCOBIX[2:0] | FCCOB Parameters |
|---|---|
| 011 | Key 2 |
| 100 | Key 3 |

Upon clearing CCIF to launch the Verify Backdoor Access Key command, the Memory Controller will check the FSEC KEYEN bits to verify that this command is enabled. If not enabled, the Memory Controller sets the ACCERR bit in the FSTAT register and terminates. If the command is enabled, the Memory Controller compares the key provided in FCCOB to the backdoor comparison key in the Flash configuration field with Key 0 compared to 0x7F_FF00, etc. If the backdoor keys match, security will be released. If the backdoor keys do not match, security is not released and all future attempts to execute the Verify Backdoor Access Key command are aborted (set ACCERR) until a reset occurs. The CCIF flag is set after the Verify Backdoor Access Key operation has completed.

**Table 24-54. Verify Backdoor Access Key Command Error Handling**

| Register | Error Bit | Error Condition |
|---|---|---|
| FSTAT | ACCERR | Set if CCOBIX[2:0] != 100 at command launch |
| | | Set if an incorrect backdoor key is supplied |
| | | Set if backdoor key access has not been enabled (KEYEN[1:0] != 10, see Section 24.3.2.2) |
| | | Set if the backdoor key has mismatched since the last reset |
| | FPVIOL | None |
| | MGSTAT1 | None |
| | MGSTAT0 | None |
| FERSTAT | EPVIOLIF | None |

### 24.4.2.12 Set User Margin Level Command

The Set User Margin Level command causes the Memory Controller to set the margin level for future read operations of a specific P-Flash or D-Flash block.

**Table 24-55. Set User Margin Level Command FCCOB Requirements**

| CCOBIX[2:0] | FCCOB Parameters | |
|---|---|---|
| 000 | 0x0D | Global address [22:16] to identify the Flash block |
| 001 | Margin level setting | |

Upon clearing CCIF to launch the Set User Margin Level command, the Memory Controller will set the user margin level for the targeted block and then set the CCIF flag.

Valid margin level settings for the Set User Margin Level command are defined in Table 24-56.

**Table 24-56. Valid Set User Margin Level Settings**

| CCOB (CCOBIX=001) | Level Description |
|---|---|
| 0x0000 | Return to Normal Level |
| 0x0001 | User Margin-1 Level(1) |
| 0x0002 | User Margin-0 Level(2) |

1. Read margin to the erased state
2. Read margin to the programmed state

**Table 24-57. Set User Margin Level Command Error Handling**

| Register | Error Bit | Error Condition |
|---|---|---|
| FSTAT | ACCERR | Set if CCOBIX[2:0] != 001 at command launch |
| | | Set if command not available in current mode (see Table 24-30) |
| | | Set if an invalid global address [22:16] is supplied(1) |
| | | Set if an invalid margin level setting is supplied |
| | FPVIOL | None |
| | MGSTAT1 | None |
| | MGSTAT0 | None |
| FERSTAT | EPVIOLIF | None |

1. As defined by the memory map for FTM256K2.

**NOTE**

User margin levels can be used to check that Flash memory contents have adequate margin for normal level read operations. If unexpected results are encountered when checking Flash memory contents at user margin levels, a potential loss of information has been detected.

### 24.4.2.13 Set Field Margin Level Command

The Set Field Margin Level command, valid in special modes only, causes the Memory Controller to set the margin level specified for future read operations of a specific P-Flash or D-Flash block.

**Table 24-58. Set Field Margin Level Command FCCOB Requirements**

| CCOBIX[2:0] | FCCOB Parameters | |
|---|---|---|
| 000 | 0x0E | Global address [22:16] to identify the Flash block |
| 001 | Margin level setting | |

Upon clearing CCIF to launch the Set Field Margin Level command, the Memory Controller will set the field margin level for the targeted block and then set the CCIF flag.

Valid margin level settings for the Set Field Margin Level command are defined in Table 24-59.

**Table 24-59. Valid Set Field Margin Level Settings**

| CCOB (CCOBIX=001) | Level Description |
|---|---|
| 0x0000 | Return to Normal Level |
| 0x0001 | User Margin-1 Level(1) |
| 0x0002 | User Margin-0 Level(2) |
| 0x0003 | Field Margin-1 Level[1] |
| 0x0004 | Field Margin-0 Level[2] |

1. Read margin to the erased state
2. Read margin to the programmed state

**Table 24-60. Set Field Margin Level Command Error Handling**

| Register | Error Bit | Error Condition |
|---|---|---|
| FSTAT | ACCERR | Set if CCOBIX[2:0] != 001 at command launch |
| | | Set if command not available in current mode (see Table 24-30) |
| | | Set if an invalid global address [22:16] is supplied(1) |
| | | Set if an invalid margin level setting is supplied |
| | FPVIOL | None |
| | MGSTAT1 | None |
| | MGSTAT0 | None |
| FERSTAT | EPVIOLIF | None |

1. As defined by the memory map for FTM256K2.

**CAUTION**

Field margin levels must only be used during verify of the initial factory programming.

**NOTE**

Field margin levels can be used to check that Flash memory contents have adequate margin for data retention at the normal level setting. If unexpected results are encountered when checking Flash memory contents at field margin levels, the Flash memory contents should be erased and reprogrammed.

### 24.4.2.14 Full Partition D-Flash Command

The Full Partition D-Flash command allows the user to allocate sectors within the D-Flash block for applications and a partition within the buffer RAM for EEPROM access. The D-Flash block consists of 128 sectors with 256 bytes per sector.

**Table 24-61. Full Partition D-Flash Command FCCOB Requirements**

| CCOBIX[2:0] | FCCOB Parameters | |
|---|---|---|
| 000 | 0x0F | Not required |
| 001 | Number of 256 byte sectors for the D-Flash user partition (DFPART) | |
| 010 | Number of 256 byte sectors for buffer RAM EEE partition (ERPART) | |

Upon clearing CCIF to launch the Full Partition D-Flash command, the following actions are taken to define a partition within the D-Flash block for direct access (DFPART) and a partition within the buffer RAM for EEE use (ERPART):

- Validate the DFPART and ERPART values provided:
  - DFPART <= 128 (maximum number of 256 byte sectors in D-Flash block)
  - ERPART <= 8 (maximum number of 256 byte sectors in buffer RAM)
  - If ERPART > 0, 128 - DFPART >= 12 (minimum number of 256 byte sectors in the D-Flash block required to support EEE)
  - If ERPART > 0, ((128-DFPART)/ERPART) >= 8 (minimum ratio of D-Flash EEE space to buffer RAM EEE space to support EEE)
- Erase the D-Flash block and the EEE nonvolatile information register
- Program DFPART to the EEE nonvolatile information register at global address 0x12_0000 (see Table 24-7)
- Program a duplicate DFPART to the EEE nonvolatile information register at global address 0x12_0002 (see Table 24-7)
- Program ERPART to the EEE nonvolatile information register at global address 0x12_0004 (see Table 24-7)
- Program a duplicate ERPART to the EEE nonvolatile information register at global address 0x12_0006 (see Table 24-7)

The D-Flash user partition will start at global address 0x10_0000. The buffer RAM EEE partition will end at global address 0x13_FFFF. After the Full Partition D-Flash operation has completed, the CCIF flag will set.

Running the Full Partition D-Flash command a second time will result in the previous partition values and the entire D-Flash memory being erased. The data value written corresponds to the number of 256 byte sectors allocated for either direct D-Flash access (DFPART) or buffer RAM EEE access (ERPART).

**Table 24-62. Full Partition D-Flash Command Error Handling**

| Register | Error Bit | Error Condition |
|---|---|---|
| FSTAT | ACCERR | Set if CCOBIX[2:0] != 010 at command launch |
| | | Set if command not available in current mode (see Table 24-30) |
| | | Set if an invalid DFPART or ERPART selection is supplied(1) |
| | FPVIOL | None |
| | MGSTAT1 | Set if any errors have been encountered during the read |
| | MGSTAT0 | Set if any non-correctable errors have been encountered during the read |
| FERSTAT | EPVIOLIF | None |

1. As defined by the maximum ERPART for FTM256K2.

### 24.4.2.15 Erase Verify D-Flash Section Command

The Erase Verify D-Flash Section command will verify that a section of code in the D-Flash user partition is erased. The Erase Verify D-Flash Section command defines the starting point of the data to be verified and the number of words.

**Table 24-63. Erase Verify D-Flash Section Command FCCOB Requirements**

| CCOBIX[2:0] | FCCOB Parameters | |
|---|---|---|
| 000 | 0x10 | Global address [22:16] to identify the D-Flash block |
| 001 | Global address [15:0] of the first word to be verified | |
| 010 | Number of words to be verified | |

Upon clearing CCIF to launch the Erase Verify D-Flash Section command, the Memory Controller will verify the selected section of D-Flash memory is erased. The CCIF flag will set after the Erase Verify D-Flash Section operation has completed.

**Table 24-64. Erase Verify D-Flash Section Command Error Handling**

| Register | Error Bit | Error Condition |
|---|---|---|
| FSTAT | ACCERR | Set if CCOBIX[2:0] != 010 at command launch |
| | | Set if command not available in current mode (see Table 24-30) |
| | | Set if an invalid global address [22:0] is supplied |
| | | Set if a misaligned word address is supplied (global address [0] != 0) |
| | | Set if the global address [22:0] points to an area of the D-Flash EEE partition |
| | | Set if the requested section breaches the end of the D-Flash block or goes into the D-Flash EEE partition |
| | FPVIOL | None |
| | MGSTAT1 | Set if any errors have been encountered during the read |
| | MGSTAT0 | Set if any non-correctable errors have been encountered during the read |
| FERSTAT | EPVIOLIF | None |

### 24.4.2.16 Program D-Flash Command

The Program D-Flash operation programs one to four previously erased words in the D-Flash user partition. The Program D-Flash operation will confirm that the targeted location(s) were successfully programmed upon completion.

**CAUTION**

A Flash word must be in the erased state before being programmed. Cumulative programming of bits within a Flash word is not allowed.

**Table 24-65. Program D-Flash Command FCCOB Requirements**

| CCOBIX[2:0] | FCCOB Parameters | |
|---|---|---|
| 000 | 0x11 | Global address [22:16] to identify the D-Flash block |
| 001 | Global address [15:0] of word to be programmed | |
| 010 | Word 0 program value | |
| 011 | Word 1 program value, if desired | |
| 100 | Word 2 program value, if desired | |
| 101 | Word 3 program value, if desired | |

Upon clearing CCIF to launch the Program D-Flash command, the user-supplied words will be transferred to the Memory Controller and be programmed. The CCOBIX index value at Program D-Flash command launch determines how many words will be programmed in the D-Flash block. No protection checks are made in the Program D-Flash operation on the D-Flash block, only access error checks. The CCIF flag is set when the operation has completed.

**Table 24-66. Program D-Flash Command Error Handling**

| Register | Error Bit | Error Condition |
|---|---|---|
| FSTAT | ACCERR | Set if CCOBIX[2:0] < 010 at command launch |
| | | Set if CCOBIX[2:0] > 101 at command launch |
| | | Set if command not available in current mode (see Table 24-30) |
| | | Set if an invalid global address [22:0] is supplied |
| | | Set if a misaligned word address is supplied (global address [0] != 0) |
| | | Set if the global address [22:0] points to an area in the D-Flash EEE partition |
| | | Set if the requested group of words breaches the end of the D-Flash block or goes into the D-Flash EEE partition |
| | FPVIOL | None |
| | MGSTAT1 | Set if any errors have been encountered during the verify operation |
| | MGSTAT0 | Set if any non-correctable errors have been encountered during the verify operation |
| FERSTAT | EPVIOLIF | None |

### 24.4.2.17 Erase D-Flash Sector Command

The Erase D-Flash Sector operation will erase all addresses in a sector of the D-Flash user partition.

**Table 24-67. Erase D-Flash Sector Command FCCOB Requirements**

| CCOBIX[2:0] | FCCOB Parameters | |
|---|---|---|
| 000 | 0x12 | Global address [22:16] to identify D-Flash block |
| 001 | Global address [15:0] anywhere within the sector to be erased. See Section 24.1.2.2 for D-Flash sector size. | |

Upon clearing CCIF to launch the Erase D-Flash Sector command, the Memory Controller will erase the selected Flash sector and verify that it is erased. The CCIF flag will set after the Erase D-Flash Sector operation has completed.

**Table 24-68. Erase D-Flash Sector Command Error Handling**

| Register | Error Bit | Error Condition |
|---|---|---|
| FSTAT | ACCERR | Set if CCOBIX[2:0] != 001 at command launch |
| | | Set if command not available in current mode (see Table 24-30) |
| | | Set if an invalid global address [22:0] is supplied |
| | | Set if a misaligned word address is supplied (global address [0] != 0) |
| | | Set if the global address [22:0] points to the D-Flash EEE partition |
| | FPVIOL | None |
| | MGSTAT1 | Set if any errors have been encountered during the verify operation |
| | MGSTAT0 | Set if any non-correctable errors have been encountered during the verify operation |
| FERSTAT | EPVIOLIF | None |

### 24.4.2.18 Enable EEPROM Emulation Command

The Enable EEPROM Emulation command causes the Memory Controller to enable EEE activity. EEE activity is disabled after any reset.

**Table 24-69. Enable EEPROM Emulation Command FCCOB Requirements**

| CCOBIX[2:0] | FCCOB Parameters | |
|---|---|---|
| 000 | 0x13 | Not required |

Upon clearing CCIF to launch the Enable EEPROM Emulation command, the CCIF flag will set after the Memory Controller enables EEE operations using the contents of the EEE tag RAM and tag counter. The Full Partition D-Flash or the Partition D-Flash command must be run prior to launching the Enable EEPROM Emulation command.

**Table 24-70. Enable EEPROM Emulation Command Error Handling**

| Register | Error Bit | Error Condition |
|---|---|---|
| FSTAT | ACCERR | Set if CCOBIX[2:0] != 000 at command launch |
| | | Set if Full Partition D-Flash or Partition D-Flash command not previously run |
| | FPVIOL | None |
| | MGSTAT1 | None |
| | MGSTAT0 | None |
| FERSTAT | EPVIOLIF | None |

### 24.4.2.19 Disable EEPROM Emulation Command

The Disable EEPROM Emulation command causes the Memory Controller to suspend current EEE activity.

**Table 24-71. Disable EEPROM Emulation Command FCCOB Requirements**

| CCOBIX[2:0] | FCCOB Parameters | |
|---|---|---|
| 000 | 0x14 | Not required |

Upon clearing CCIF to launch the Disable EEPROM Emulation command, the Memory Controller will halt EEE operations at the next convenient point without clearing the EEE tag RAM or tag counter before setting the CCIF flag.

**Table 24-72. Disable EEPROM Emulation Command Error Handling**

| Register | Error Bit | Error Condition |
|---|---|---|
| FSTAT | ACCERR | Set if CCOBIX[2:0] != 000 at command launch |
| | | Set if Full Partition D-Flash or Partition D-Flash command not previously run |
| | FPVIOL | None |
| | MGSTAT1 | None |
| | MGSTAT0 | None |
| FERSTAT | EPVIOLIF | None |

### 24.4.2.20 EEPROM Emulation Query Command

The EEPROM Emulation Query command returns EEE partition and status variables.

**Table 24-73. EEPROM Emulation Query Command FCCOB Requirements**

| CCOBIX[2:0] | FCCOB Parameters | |
|---|---|---|
| 000 | 0x15 | Not required |
| 001 | Return DFPART | |
| 010 | Return ERPART | |
| 011 | Return ECOUNT(1) | |
| 100 | Return Dead Sector Count | Return Ready Sector Count |

1. Indicates sector erase count

Upon clearing CCIF to launch the EEPROM Emulation Query command, the CCIF flag will set after the EEE partition and status variables are stored in the FCCOBIX register.If the Emulation Query command is executed prior to partitioning (Partition D-Flash Command Section 24.4.2.14), the following reset values are returned: DFPART = 0x_FFFF, ERPART = 0x_FFFF, ECOUNT = 0x_FFFF, Dead Sector Count = 0x_00, Ready Sector Count = 0x_00.

**Table 24-74. EEPROM Emulation Query Command Error Handling**

| Register | Error Bit | Error Condition |
|---|---|---|
| FSTAT | ACCERR | Set if CCOBIX[2:0] != 000 at command launch |
| | | Set if command not available in current mode (see Table 24-30) |
| | FPVIOL | None |
| | MGSTAT1 | None |
| | MGSTAT0 | None |
| FERSTAT | EPVIOLIF | None |

### 24.4.2.21 Partition D-Flash Command

The Partition D-Flash command allows the user to allocate sectors within the D-Flash block for applications and a partition within the buffer RAM for EEPROM access. The D-Flash block consists of 32 sectors with 256 bytes per sector. The Erase All Blocks command must be run prior to launching the Partition D-Flash command.

**Table 24-75. Partition D-Flash Command FCCOB Requirements**

| CCOBIX[2:0] | FCCOB Parameters | |
|---|---|---|
| 000 | 0x20 | Not required |
| 001 | Number of 256 byte sectors for the D-Flash user partition (DFPART) | |
| 010 | Number of 256 byte sectors for buffer RAM EEE partition (ERPART) | |

Upon clearing CCIF to launch the Partition D-Flash command, the following actions are taken to define a partition within the D-Flash block for direct access (DFPART) and a partition within the buffer RAM for EEE use (ERPART):

- Validate the DFPART and ERPART values provided:
  - DFPART <= 128 (maximum number of 256 byte sectors in D-Flash block)
  - ERPART <= 8 (maximum number of 256 byte sectors in buffer RAM)
  - If ERPART > 0, 128 - DFPART >= 12 (minimum number of 256 byte sectors in the D-Flash block required to support EEE)
  - If ERPART > 0, ((128-DFPART)/ERPART) >= 8 (minimum ratio of D-Flash EEE space to buffer RAM EEE space to support EEE)
- Erase verify the D-Flash block and the EEE nonvolatile information register
- Program DFPART to the EEE nonvolatile information register at global address 0x12_0000 (see Table 24-7)
- Program a duplicate DFPART to the EEE nonvolatile information register at global address 0x12_0002 (see Table 24-7)

- Program ERPART to the EEE nonvolatile information register at global address 0x12_0004 (see Table 24-7)
- Program a duplicate ERPART to the EEE nonvolatile information register at global address 0x12_0006 (see Table 24-7)

The D-Flash user partition will start at global address 0x10_0000. The buffer RAM EEE partition will end at global address 0x13_FFFF. After the Partition D-Flash operation has completed, the CCIF flag will set.

Running the Partition D-Flash command a second time will result in the ACCERR bit within the FSTAT register being set. The data value written corresponds to the number of 256 byte sectors allocated for either direct D-Flash access (DFPART) or buffer RAM EEE access (ERPART).

**Table 24-76. Partition D-Flash Command Error Handling**

| Register | Error Bit | Error Condition |
|---|---|---|
| FSTAT | ACCERR | Set if CCOBIX[2:0] != 010 at command launch |
| | | Set if command not available in current mode (see Table 24-30) |
| | | Set if partitions have already been defined |
| | | Set if an invalid DFPART or ERPART selection is supplied(1) |
| | FPVIOL | None |
| | MGSTAT1 | Set if any errors have been encountered during the read |
| | MGSTAT0 | Set if any non-correctable errors have been encountered during the read |
| FERSTAT | EPVIOLIF | None |

1. As defined by the maximum ERPART for FTM256K2.

### 24.4.3 Interrupts

The Flash module can generate an interrupt when a Flash command operation has completed or when a Flash command operation has detected an EEE error or an ECC fault.

**Table 24-77. Flash Interrupt Sources**

| Interrupt Source | Interrupt Flag | Local Enable | Global (CCR) Mask |
|---|---|---|---|
| Flash Command Complete | CCIF (FSTAT register) | CCIE (FCNFG register) | I Bit |
| Flash EEE Erase Error | ERSERIF (FERSTAT register) | ERSERIE (FERCNFG register) | I Bit |
| Flash EEE Program Error | PGMERIF (FERSTAT register) | PGMERIE (FERCNFG register) | I Bit |
| Flash EEE Protection Violation | EPVIOLIF (FERSTAT register) | EPVIOLIE (FERCNFG register) | I Bit |
| Flash EEE Error Type 1 Violation | ERSVIF1 (FERSTAT register) | ERSVIE1 (FERCNFG register) | I Bit |
| Flash EEE Error Type 0 Violation | ERSVIF0 (FERSTAT register) | ERSVIE0 (FERCNFG register) | I Bit |
| ECC Double Bit Fault on Flash Read | DFDIF (FERSTAT register) | DFDIE (FERCNFG register) | I Bit |
| ECC Single Bit Fault on Flash Read | SFDIF (FERSTAT register) | SFDIE (FERCNFG register) | I Bit |

**NOTE**

Vector addresses and their relative interrupt priority are determined at the MCU level.

#### 24.4.3.1 Description of Flash Interrupt Operation

The Flash module uses the CCIF flag in combination with the CCIE interrupt enable bit to generate the Flash command interrupt request. The Flash module uses the ERSEIF, PGMEIF, EPVIOLIF, ERSVIF1, ERSVIF0, DFDIF and SFDIF flags in combination with the ERSEIE, PGMEIE, EPVIOLIE, ERSVIE1, ERSVIE0, DFDIE and SFDIE interrupt enable bits to generate the Flash error interrupt request. For a detailed description of the register bits involved, refer to Section 24.3.2.5, “Flash Configuration Register (FCNFG)”, Section 24.3.2.6, “Flash Error Configuration Register (FERCNFG)”, Section 24.3.2.7, “Flash Status Register (FSTAT)”, and Section 24.3.2.8, “Flash Error Status Register (FERSTAT)”.

The logic used for generating the Flash module interrupts is shown in Figure 24-27.

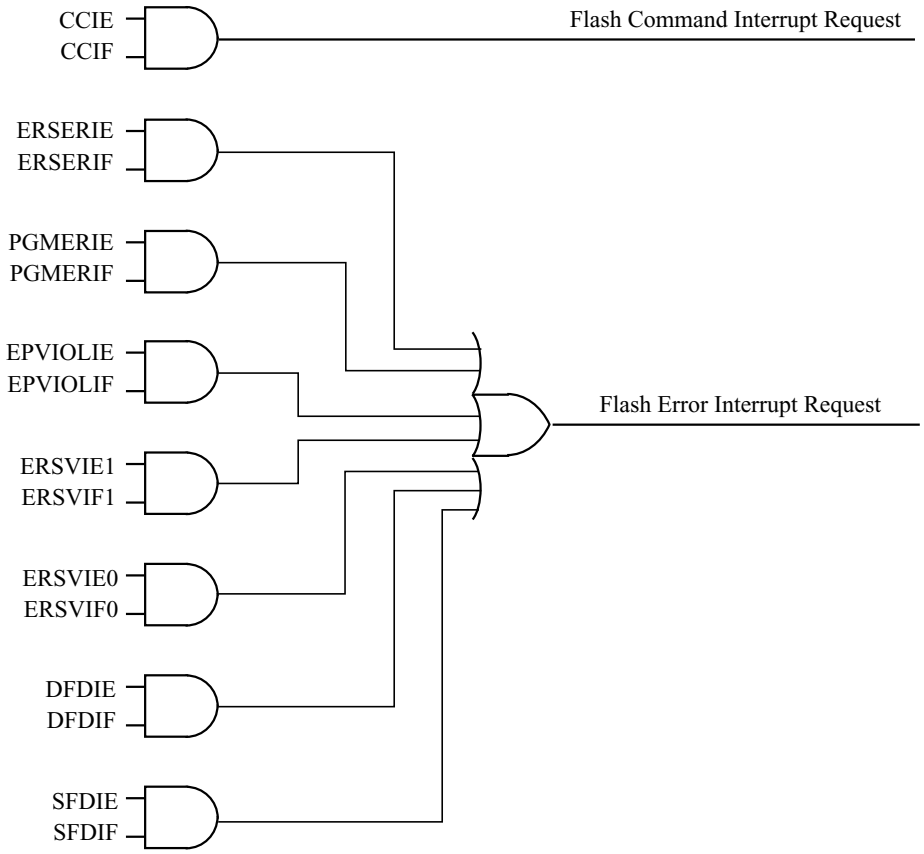



**Figure 24-27. Flash Module Interrupts Implementation**

### 24.4.4 Wait Mode

The Flash module is not affected if the MCU enters wait mode. The Flash module can recover the MCU from wait via the CCIF interrupt (see Section 24.4.3, “Interrupts”).

### 24.4.5 Stop Mode

If a Flash command is active (CCIF = 0) or an EE-Emulation operation is pending when the MCU requests stop mode, the current Flash operation will be completed before the CPU is allowed to enter stop mode.

## 24.5 Security

The Flash module provides security information to the MCU. The Flash security state is defined by the SEC bits of the FSEC register (see Table 24-12). During reset, the Flash module initializes the FSEC register using data read from the security byte of the Flash configuration field at global address 0x7F_FF0F.

The security state out of reset can be permanently changed by programming the security byte of the Flash configuration field. This assumes that you are starting from a mode where the necessary P-Flash erase and program commands are available and that the upper region of the P-Flash is unprotected. If the Flash security byte is successfully programmed, its new value will take affect after the next MCU reset.

The following subsections describe these security-related subjects:

- Unsecuring the MCU using Backdoor Key Access
- Unsecuring the MCU in Special Single Chip Mode using BDM
- Mode and Security Effects on Flash Command Availability

### 24.5.1 Unsecuring the MCU using Backdoor Key Access

The MCU may be unsecured by using the backdoor key access feature which requires knowledge of the contents of the backdoor keys (four 16-bit words programmed at addresses 0x7F_FF00–0x7F_FF07). If the KEYEN[1:0] bits are in the enabled state (see Section 24.3.2.2), the Verify Backdoor Access Key command (see Section 24.4.2.11) allows the user to present four prospective keys for comparison to the keys stored in the Flash memory via the Memory Controller. If the keys presented in the Verify Backdoor Access Key command match the backdoor keys stored in the Flash memory, the SEC bits in the FSEC register (see Table 24-12) will be changed to unsecure the MCU. Key values of 0x0000 and 0xFFFF are not permitted as backdoor keys. While the Verify Backdoor Access Key command is active, P-Flash block 0 will not be available for read access and will return invalid data.

The user code stored in the P-Flash memory must have a method of receiving the backdoor keys from an external stimulus. This external stimulus would typically be through one of the on-chip serial ports.

If the KEYEN[1:0] bits are in the enabled state (see Section 24.3.2.2), the MCU can be unsecured by the backdoor key access sequence described below:

1. Follow the command sequence for the Verify Backdoor Access Key command as explained in Section 24.4.2.11
2. If the Verify Backdoor Access Key command is successful, the MCU is unsecured and the SEC[1:0] bits in the FSEC register are forced to the unsecure state of 10

The Verify Backdoor Access Key command is monitored by the Memory Controller and an illegal key will prohibit future use of the Verify Backdoor Access Key command. A reset of the MCU is the only method to re-enable the Verify Backdoor Access Key command.

After the backdoor keys have been correctly matched, the MCU will be unsecured. After the MCU is unsecured, the sector containing the Flash security byte can be erased and the Flash security byte can be reprogrammed to the unsecure state, if desired.

In the unsecure state, the user has full control of the contents of the backdoor keys by programming addresses 0x7F_FF00–0x7F_FF07 in the Flash configuration field.

The security as defined in the Flash security byte (0x7F_FF0F) is not changed by using the Verify Backdoor Access Key command sequence. The backdoor keys stored in addresses 0x7F_FF00–0x7F_FF07 are unaffected by the Verify Backdoor Access Key command sequence. After the next reset of the MCU, the security state of the Flash module is determined by the Flash security byte

(0x7F_FF0F). The Verify Backdoor Access Key command sequence has no effect on the program and erase protections defined in the Flash protection register, FPROT.

### 24.5.2 Unsecuring the MCU in Special Single Chip Mode using BDM

The MCU can be unsecured in special single chip mode by erasing the P-Flash and D-Flash memory by one of the following methods:

- Reset the MCU into special single chip mode, delay while the erase test is performed by the BDM, send BDM commands to disable protection in the P-Flash and D-Flash memory, and execute the Erase All Blocks command write sequence to erase the P-Flash and D-Flash memory.
- Reset the MCU into special expanded wide mode, disable protection in the P-Flash and D-Flash memory and run code from external memory to execute the Erase All Blocks command write sequence to erase the P-Flash and D-Flash memory.

After the CCIF flag sets to indicate that the Erase All Blocks operation has completed, reset the MCU into special single chip mode. The BDM will execute the Erase Verify All Blocks command write sequence to verify that the P-Flash and D-Flash memory is erased. If the P-Flash and D-Flash memory are verified as erased the MCU will be unsecured. All BDM commands will be enabled and the Flash security byte may be programmed to the unsecure state by the following method:

- Send BDM commands to execute a ‘Program P-Flash’ command sequence to program the Flash security byte to the unsecured state and reset the MCU.

### 24.5.3 Mode and Security Effects on Flash Command Availability

The availability of Flash module commands depends on the MCU operating mode and security state as shown in Table 24-30.

## 24.6 Initialization

On each system reset the Flash module executes a reset sequence which establishes initial values for the Flash Block Configuration Parameters, the FPROT and DFPROT protection registers, and the FOPT and FSEC registers. The Flash module reverts to built-in default values that leave the module in a fully protected and secured state if errors are encountered during execution of the reset sequence. If a double bit fault is detected during the reset sequence, both MGSTAT bits in the FSTAT register will be set. The ACCERR bit in the FSTAT register is set if errors are encountered while initializing the EEE buffer ram during the reset sequence.

CCIF remains clear throughout the reset sequence. The Flash module holds off all CPU access for the initial portion of the reset sequence. While Flash reads are possible when the hold is removed, writes to the FCCOBIX, FCCOBHI, and FCCOBLO registers are ignored to prevent command activity while the Memory Controller remains busy. Completion of the reset sequence is marked by setting CCIF high which enables writes to the FCCOBIX, FCCOBHI, and FCCOBLO registers to launch any available Flash command.

If a reset occurs while any Flash command is in progress, that command will be immediately aborted. The state of the word being programmed or the sector/block being erased is not guaranteed.

# Chapter 25
# 256 KByte Flash Module (S12XFTM256K2V1)

**Table 25-1. Revision History**

| Revision Number | Revision Date | Sections Affected | Description of Changes |
|---|---|---|---|
| V01.08 | 14 Nov 2007 | 25.5.2/25-951<br><br>25.4.2/25-927<br><br><br>25.4.2.8/25-933 | - Changed terminology from ‘word program’ to “Program P-Flash’ in the BDM unsecuring description, Section 25.5.2<br>- Added requirement that user not write any Flash module register during execution of commands ‘Erase All Blocks’, Section 25.4.2.8, and ‘Unsecure Flash’, Section 25.4.2.11<br>- Added statement that security is released upon successful completion of command ‘Erase All Blocks’, Section 25.4.2.8 |
| V01.09 | 19 Dec 2007 | 25.4.2.5/25-930<br>25.4.2/25-927<br><br>25.3.1/25-896 | - Corrected Error Handling table for Load Data Field command<br>- Corrected Error Handling table for Full Partition D-Flash, Partition D-Flash, and EEPROM Emulation Query commands<br>- Corrected P-Flash IFR Accessibility table |
| V01.10 | 25 Sep 2009 | 25.1/25-891<br>25.3.2.1/25-903<br>25.4.2.4/25-930<br><br>25.4.2.7/25-932<br><br>25.4.2.12/25-936<br><br>25.4.2.12/25-936<br>25.4.2.12/25-936<br>25.4.2.20/25-945<br><br><br>25.3.2/25-901<br>25.3.2.1/25-903<br>25.4.1.2/25-922<br>25.6/25-951 | - Clarify single bit fault correction for P-Flash phrase<br>- Expand FDIV vs OSCCLK Frequency table<br>- Add statement concerning code runaway when executing Read Once command from Flash block containing associated fields<br>- Add statement concerning code runaway when executing Program Once command from Flash block containing associated fields<br>- Add statement concerning code runaway when executing Verify Backdoor Access Key command from Flash block containing associated fields<br>- Relate Key 0 to associated Backdoor Comparison Key address<br>- Change “power down reset” to “reset”<br>- Add ACCERR condition for Disable EEPROM Emulation command<br>The following changes were made to clarify module behavior related to Flash register access during reset sequence and while Flash commands are active:<br>- Add caution concerning register writes while command is active<br>- Writes to FCLKDIV are allowed during reset sequence while CCIF is clear<br>- Add caution concerning register writes while command is active<br>- Writes to FCCOBIX, FCCOBHI, FCCOBLO registers are ignored during reset sequence |

## 25.1 Introduction

The FTM256K2 module implements the following:

- 256 Kbytes of P-Flash (Program Flash) memory, consisting of 2 physical Flash blocks, intended primarily for nonvolatile code storage

- 32 Kbytes of D-Flash (Data Flash) memory, consisting of 1 physical Flash block, that can be used as nonvolatile storage to support the built-in hardware scheme for emulated EEPROM, as basic Flash memory primarily intended for nonvolatile data storage, or as a combination of both
- 4 Kbytes of buffer RAM, consisting of 1 physical RAM block, that can be used as emulated EEPROM using a built-in hardware scheme, as basic RAM, or as a combination of both

The Flash memory is ideal for single-supply applications allowing for field reprogramming without requiring external high voltage sources for program or erase operations. The Flash module includes a memory controller that executes commands to modify Flash memory contents or configure module resources for emulated EEPROM operation. The user interface to the memory controller consists of the indexed Flash Common Command Object (FCCOB) register which is written to with the command, global address, data, and any required command parameters. The memory controller must complete the execution of a command before the FCCOB register can be written to with a new command.

**CAUTION**

A Flash word or phrase must be in the erased state before being programmed. Cumulative programming of bits within a Flash word or phrase is not allowed.

The RAM and Flash memory may be read as bytes, aligned words, or misaligned words. Read access time is one bus cycle for bytes and aligned words, and two bus cycles for misaligned words. For Flash memory, an erased bit reads 1 and a programmed bit reads 0.

It is not possible to read from a Flash block while any command is executing on that specific Flash block. It is possible to read from a Flash block while a command is executing on a different Flash block.

Both P-Flash and D-Flash memories are implemented with Error Correction Codes (ECC) that can resolve single bit faults and detect double bit faults. For P-Flash memory, the ECC implementation requires that programming be done on an aligned 8 byte basis (a Flash phrase). Since P-Flash memory is always read by phrase, only one single bit fault in the phrase containing the byte or word accessed will be corrected.

### 25.1.1 Glossary

**Buffer RAM** — The buffer RAM constitutes the volatile memory store required for EEE. Memory space in the buffer RAM not required for EEE can be partitioned to provide volatile memory space for applications.

**Command Write Sequence** — An MCU instruction sequence to execute built-in algorithms (including program and erase) on the Flash memory.

**D-Flash Memory** — The D-Flash memory constitutes the nonvolatile memory store required for EEE. Memory space in the D-Flash memory not required for EEE can be partitioned to provide nonvolatile memory space for applications.

**D-Flash Sector** — The D-Flash sector is the smallest portion of the D-Flash memory that can be erased. The D-Flash sector consists of four 64 byte rows for a total of 256 bytes.

**EEE (Emulated EEPROM)** — A method to emulate the small sector size features and endurance characteristics associated with an EEPROM.

**EEE IFR** — Nonvolatile information register located in the D-Flash block that contains data required to partition the D-Flash memory and buffer RAM for EEE. The EEE IFR is visible in the global memory map by setting the EEEIFRON bit in the MMCCTL1 register.

**NVM Command Mode** — An NVM mode using the CPU to setup the FCCOB register to pass parameters required for Flash command execution.

**Phrase** — An aligned group of four 16-bit words within the P-Flash memory. Each phrase includes eight ECC bits for single bit fault correction and double bit fault detection within the phrase.

**P-Flash Memory** — The P-Flash memory constitutes the main nonvolatile memory store for applications.

**P-Flash Sector** — The P-Flash sector is the smallest portion of the P-Flash memory that can be erased. Each P-Flash sector contains 1024 bytes.

**Program IFR** — Nonvolatile information register located in the P-Flash block that contains the Device ID, Version ID, and the Program Once field. The Program IFR is visible in the global memory map by setting the PGMIFRON bit in the MMCCTL1 register.

## 25.1.2 Features

### 25.1.2.1 P-Flash Features

- 256 Kbytes of P-Flash memory composed of two 128 Kbyte Flash blocks. The 128 Kbyte Flash blocks are each divided into 128 sectors of 1024 bytes.
- Single bit fault correction and double bit fault detection within a 64-bit phrase during read operations
- Automated program and erase algorithm with verify and generation of ECC parity bits
- Fast sector erase and phrase program operation
- Ability to program up to one phrase in each P-Flash block simultaneously
- Flexible protection scheme to prevent accidental program or erase of P-Flash memory

### 25.1.2.2 D-Flash Features

- Up to 32 Kbytes of D-Flash memory with 256 byte sectors for user access
- Dedicated commands to control access to the D-Flash memory over EEE operation
- Single bit fault correction and double bit fault detection within a word during read operations
- Automated program and erase algorithm with verify and generation of ECC parity bits
- Fast sector erase and word program operation
- Ability to program up to four words in a burst sequence

#### 25.1.2.3 Emulated EEPROM Features

- Up to 4 Kbytes of emulated EEPROM (EEE) accessible as 4 Kbytes of RAM
- Flexible protection scheme to prevent accidental program or erase of data
- Automatic EEE file handling using an internal Memory Controller
- Automatic transfer of valid EEE data from D-Flash memory to buffer RAM on reset
- Ability to monitor the number of outstanding EEE related buffer RAM words left to be programmed into D-Flash memory
- Ability to disable EEE operation and allow priority access to the D-Flash memory
- Ability to cancel all pending EEE operations and allow priority access to the D-Flash memory

#### 25.1.2.4 User Buffer RAM Features

- Up to 4 Kbytes of RAM for user access

#### 25.1.2.5 Other Flash Module Features

- No external high-voltage power supply required for Flash memory program and erase operations
- Interrupt generation on Flash command completion and Flash error detection
- Security mechanism to prevent unauthorized access to the Flash memory

### 25.1.3 Block Diagram

The block diagram of the Flash module is shown in Figure 25-1.

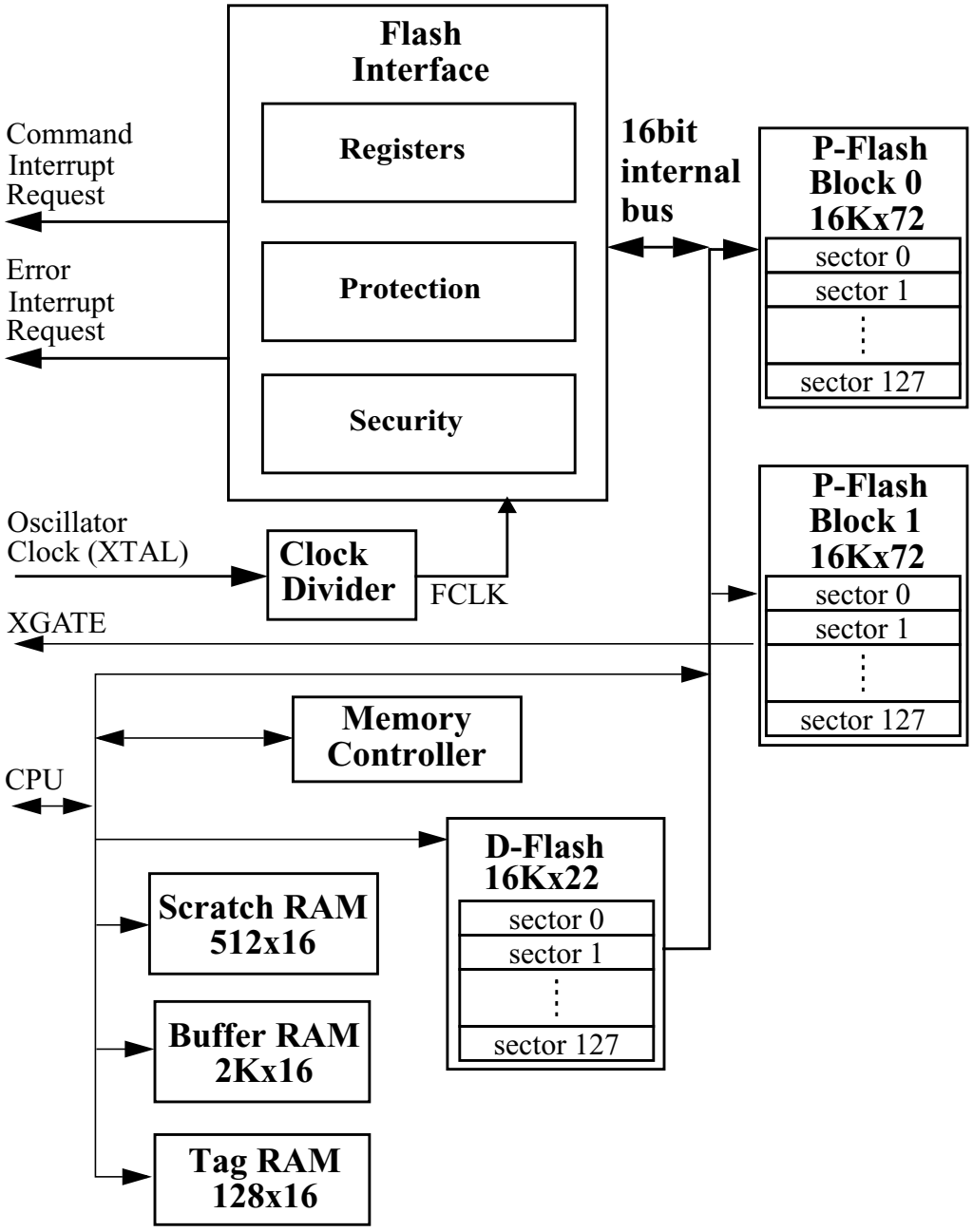



**Figure 25-1. FTM256K2 Block Diagram**

## 25.2 External Signal Description

The Flash module contains no signals that connect off-chip.

## 25.3 Memory Map and Registers

This section describes the memory map and registers for the Flash module. Read data from unimplemented memory space in the Flash module is undefined. Write access to unimplemented or reserved memory space in the Flash module will be ignored by the Flash module.

### 25.3.1 Module Memory Map

The S12X architecture places the P-Flash memory between global addresses 0x78_0000 and 0x7F_FFFF as shown in Table 25-2. The P-Flash memory map is shown in Figure 25-2.

**Table 25-2. P-Flash Memory Addressing**

| Global Address | Size (Bytes) | Description |
|---|---|---|
| 0x7E_0000 – 0x7F_FFFF | 128 K | P-Flash Block 0<br>Contains Flash Configuration Field (see Table 25-3) |
| 0x7A_0000 – 0x7D_FFFF | 256 K | No P-Flash Memory |
| 0x78_0000 – 0x79_FFFF | 128 K | P-Flash Block 1 |

The FPROT register, described in Section 25.3.2.9, can be set to protect regions in the Flash memory from accidental program or erase. Three separate memory regions, one growing upward from global address 0x7F_8000 in the Flash memory (called the lower region), one growing downward from global address 0x7F_FFFF in the Flash memory (called the higher region), and the remaining addresses in the Flash memory, can be activated for protection. The Flash memory addresses covered by these protectable regions are shown in the P-Flash memory map. The higher address region is mainly targeted to hold the boot loader code since it covers the vector space. Default protection settings as well as security information that allows the MCU to restrict access to the Flash module are stored in the Flash configuration field as described in Table 25-3.

**Table 25-3. Flash Configuration Field(1)**

| Global Address | Size (Bytes) | Description |
|---|---|---|
| 0x7F_FF00 – 0x7F_FF07 | 8 | Backdoor Comparison Key<br>Refer to Section 25.4.2.12, “Verify Backdoor Access Key Command,” and Section 25.5.1, “Unsecuring the MCU using Backdoor Key Access” |
| 0x7F_FF08 – 0x7F_FF0B(2) | 4 | Reserved |
| 0x7F_FF0C[2] | 1 | P-Flash Protection byte.<br>Refer to Section 25.3.2.9, “P-Flash Protection Register (FPROT)” |
| 0x7F_FF0D[2] | 1 | EEE Protection byte<br>Refer to Section 25.3.2.10, “EEE Protection Register (EPROT)” |
| 0x7F_FF0E[2] | 1 | Flash Nonvolatile byte<br>Refer to Section 25.3.2.14, “Flash Option Register (FOPT)” |

**Table 25-3. Flash Configuration Field[1]**

| Global Address | Size (Bytes) | Description |
|---|---|---|
| 0x7F_FF0F[2] | 1 | Flash Security byte<br>Refer to Section 25.3.2.2, “Flash Security Register (FSEC)” |

1. Older versions may have swapped protection byte addresses
2. 0x7FF08 - 0x7F_FF0F form a Flash phrase and must be programmed in a single command write sequence. Each byte in the 0x7F_FF08 - 0x7F_FF0B reserved field should be programmed to 0xFF.

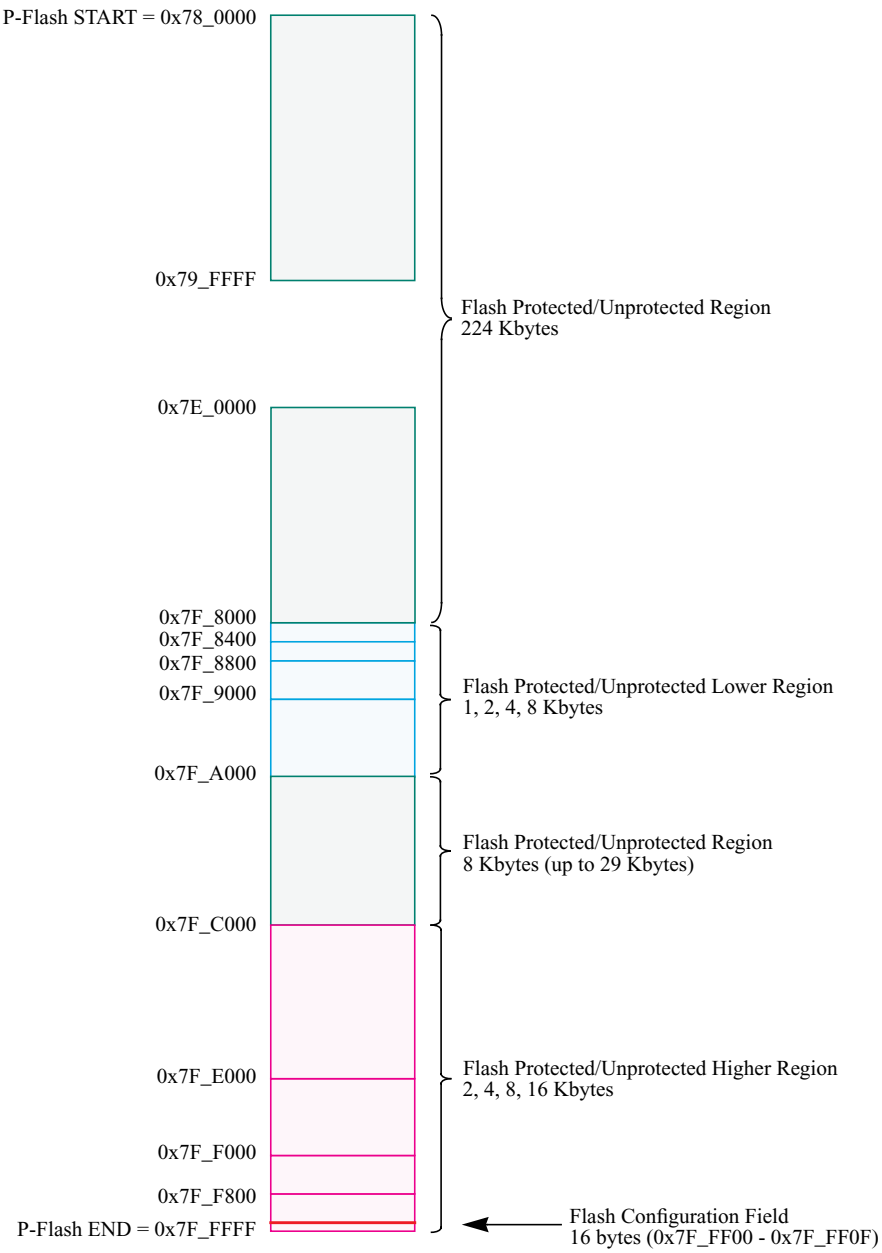



**Figure 25-2. P-Flash Memory Map**

**Table 25-4. Program IFR Fields**

| Global Address (PGMIFRON) | Size (Bytes) | Field Description |
|---|---|---|
| 0x40_0000 – 0x40_0007 | 8 | Device ID |
| 0x40_0008 – 0x40_00E7 | 224 | Reserved |
| 0x40_00E8 – 0x40_00E9 | 2 | Version ID |
| 0x40_00EA – 0x40_00FF | 22 | Reserved |
| 0x40_0100 – 0x40_013F | 64 | Program Once Field<br>Refer to Section 25.4.2.7, “Program Once Command” |
| 0x40_0140 – 0x40_01FF | 192 | Reserved |

**Table 25-5. P-Flash IFR Accessibility**

| Global Address (PGMIFRON) | Size (Bytes) | Accessed From |
|---|---|---|
| 0x40_0000 – 0x40_01FF | 512 | XBUS0 (PBLK0)(1) |
| 0x40_0200 – 0x40_03FF | 512 | Unimplemented |
| 0x40_0400 – 0x40_05FF | 512 | Unimplemented |
| 0x40_0600 – 0x40_07FF | 512 | XBUS1 (PBLK1) |

1. Refer to Table 25-4 for more details.

**Table 25-6. EEE Resource Fields**

| Global Address | Size (Bytes) | Description |
|---|---|---|
| 0x10_0000 – 0x10_7FFF | 32,768 | D-Flash Memory (User and EEE) |
| 0x10_8000 – 0x11_FFFF | 98,304 | Reserved |
| 0x12_0000 – 0x12_007F | 128 | EEE Nonvolatile Information Register (EEEIFRON(1) = 1) |
| 0x12_0080 – 0x12_0FFF | 3,968 | Reserved |
| 0x12_1000 – 0x12_1EFF | 3,840 | Reserved |
| 0x12_1F00 – 0x12_1FFF | 256 | EEE Tag RAM (TMGRAMON[1] = 1) |
| 0x12_2000 – 0x12_3BFF | 7,168 | Reserved |
| 0x12_3C00 – 0x12_3FFF | 1,024 | Memory Controller Scratch RAM (TMGRAMON[1] = 1) |
| 0x12_4000 – 0x12_DFFF | 40,960 | Reserved |
| 0x12_E000 – 0x12_FFFF | 8,192 | Reserved |
| 0x13_0000 – 0x13_EFFF | 61,440 | Reserved |
| 0x13_F000 – 0x13_FFFF | 4,096 | Buffer RAM (User and EEE) |

1. MMCCTL1 register bit

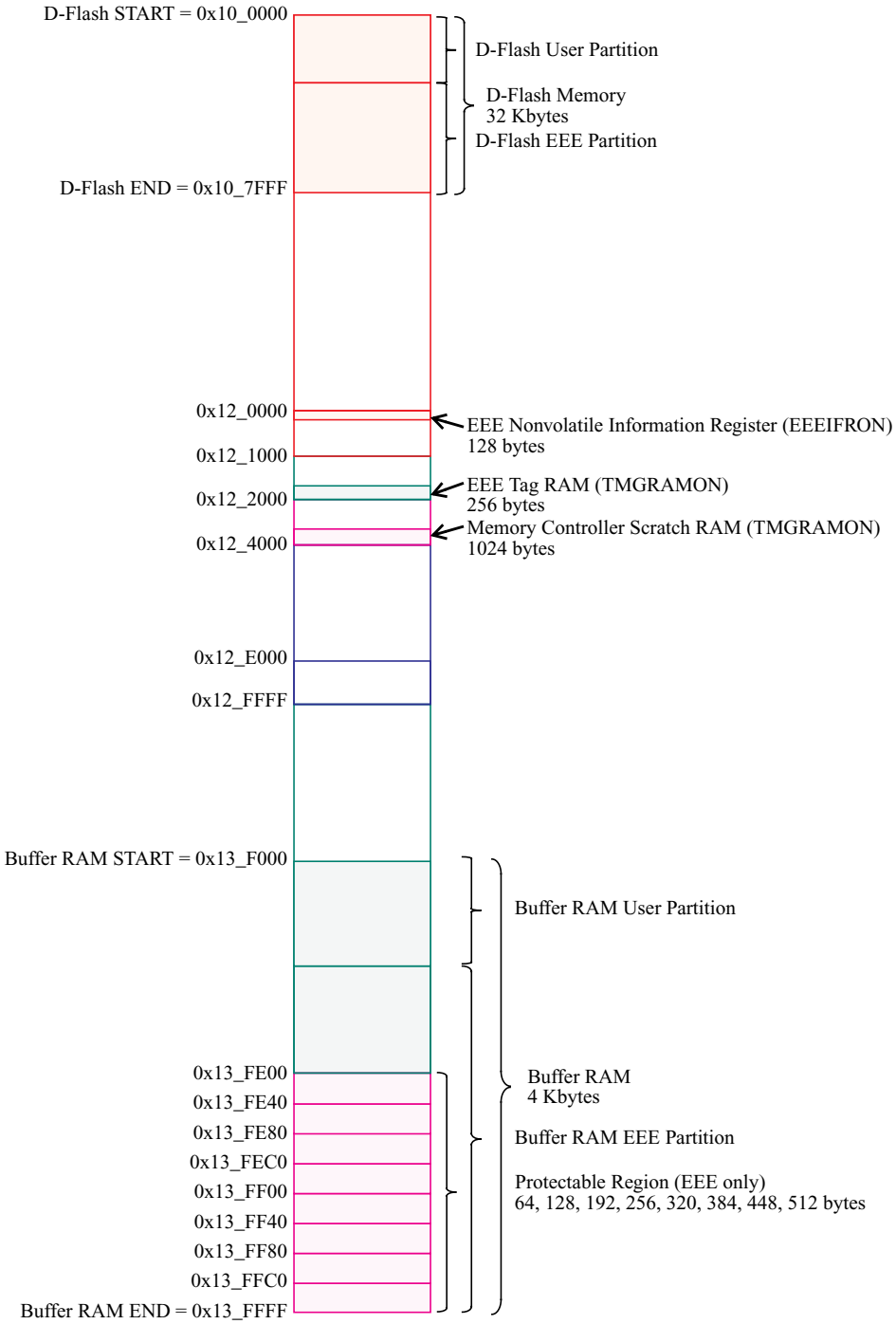



**Figure 25-3. EEE Resource Memory Map**

The Full Partition D-Flash command (see Section 25.4.2.15) is used to program the EEE nonvolatile information register fields where address 0x12_0000 defines the D-Flash partition for user access and address 0x12_0004 defines the buffer RAM partition for EEE operations.

**Table 25-7. EEE Nonvolatile Information Register Fields**

| Global Address (EEEIFRON) | Size (Bytes) | Description |
|---|---|---|
| 0x12_0000 – 0x12_0001 | 2 | D-Flash User Partition (DFPART)<br>Refer to Section 25.4.2.15, “Full Partition D-Flash Command” |
| 0x12_0002 – 0x12_0003 | 2 | D-Flash User Partition (duplicate(1)) |
| 0x12_0004 – 0x12_0005 | 2 | Buffer RAM EEE Partition (ERPART)<br>Refer to Section 25.4.2.15, “Full Partition D-Flash Command” |
| 0x12_0006 – 0x12_0007 | 2 | Buffer RAM EEE Partition (duplicate1) |
| 0x12_0008 – 0x12_007F | 120 | Reserved |

1. Duplicate value used if primary value generates a double bit fault when read during the reset sequence.

## 25.3.2 Register Descriptions

The Flash module contains a set of 20 control and status registers located between Flash module base + 0x0000 and 0x0013. A summary of the Flash module registers is given in Figure 25-4 with detailed descriptions in the following subsections.

**CAUTION**

Writes to any Flash register must be avoided while a Flash command is active (CCIF=0) to prevent corruption of Flash register contents and Memory Controller behavior.

| Address & Name | | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|
| 0x0000 FCLKDIV | R | FDIVLD | FDIV6 | FDIV5 | FDIV4 | FDIV3 | FDIV2 | FDIV1 | FDIV0 |
| | W | | | | | | | | |
| 0x0001 FSEC | R | KEYEN1 | KEYEN0 | RNV5 | RNV4 | RNV3 | RNV2 | SEC1 | SEC0 |
| | W | | | | | | | | |
| 0x0002 FCCOBIX | R | 0 | 0 | 0 | 0 | 0 | CCOBIX2 | CCOBIX1 | CCOBIX0 |
| | W | | | | | | | | |
| 0x0003 FECCRIX | R | 0 | 0 | 0 | 0 | 0 | ECCRIX2 | ECCRIX1 | ECCRIX0 |
| | W | | | | | | | | |
| 0x0004 FCNFG | R | CCIE | 0 | 0 | IGNSF | 0 | 0 | FDFD | FSFD |
| | W | | | | | | | | |
| 0x0005 FERCNFG | R | ERSERIE | PGMERIE | 0 | EPVIOLIE | ERSVIE1 | ERSVIE0 | DFDIE | SFDIE |
| | W | | | | | | | | |

**Figure 25-4. FTM256K2 Register Summary**

| Address & Name | | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|
| 0x0006 FSTAT | R | CCIF | 0 | ACCERR | FPVIOL | MGBUSY | RSVD | MGSTAT1 | MGSTAT0 |
| | W | | | | | | | | |
| 0x0007 FERSTAT | R | ERSERIF | PGMERIF | 0 | EPVIOLIF | ERSVIF1 | ERSVIF0 | DFDIF | SFDIF |
| | W | | | | | | | | |
| 0x0008 FPROT | R | FPOPEN | RNV6 | FPHDIS | FPHS1 | FPHS0 | FPLDIS | FPLS1 | FPLS0 |
| | W | | | | | | | | |
| 0x0009 EPROT | R | EPOPEN | RNV6 | RNV5 | RNV4 | EPDIS | EPS2 | EPS1 | EPS0 |
| | W | | | | | | | | |
| 0x000A FCCOBHI | R | CCOB15 | CCOB14 | CCOB13 | CCOB12 | CCOB11 | CCOB10 | CCOB9 | CCOB8 |
| | W | | | | | | | | |
| 0x000B FCCOBLO | R | CCOB7 | CCOB6 | CCOB5 | CCOB4 | CCOB3 | CCOB2 | CCOB1 | CCOB0 |
| | W | | | | | | | | |
| 0x000C ETAGHI | R | ETAG15 | ETAG14 | ETAG13 | ETAG12 | ETAG11 | ETAG10 | ETAG9 | ETAG8 |
| | W | | | | | | | | |
| 0x000D ETAGLO | R | ETAG7 | ETAG6 | ETAG5 | ETAG4 | ETAG3 | ETAG2 | ETAG1 | ETAG0 |
| | W | | | | | | | | |
| 0x000E FECCRHI | R | ECCR15 | ECCR14 | ECCR13 | ECCR12 | ECCR11 | ECCR10 | ECCR9 | ECCR8 |
| | W | | | | | | | | |
| 0x000F FECCRLO | R | ECCR7 | ECCR6 | ECCR5 | ECCR4 | ECCR3 | ECCR2 | ECCR1 | ECCR0 |
| | W | | | | | | | | |
| 0x0010 FOPT | R | NV7 | NV6 | NV5 | NV4 | NV3 | NV2 | NV1 | NV0 |
| | W | | | | | | | | |
| 0x0011 FRSV0 | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | W | | | | | | | | |
| 0x0012 FRSV1 | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | W | | | | | | | | |
| 0x0013 FRSV2 | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | W | | | | | | | | |

**Figure 25-4. FTM256K2 Register Summary (continued)**

| Address & Name | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | = Unimplemented or Reserved | | | | | | | |

**Figure 25-4. FTM256K2 Register Summary (continued)**

### 25.3.2.1 Flash Clock Divider Register (FCLKDIV)

The FCLKDIV register is used to control timed events in program and erase algorithms.

Offset Module Base + 0x0000

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | FDIVLD | FDIV[6:0] | | | | | | |
| W | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented or Reserved

**Figure 25-5. Flash Clock Divider Register (FCLKDIV)**

All bits in the FCLKDIV register are readable, bits 6–0 are write once and bit 7 is not writable.

**Table 25-8. FCLKDIV Field Descriptions**

| Field | Description |
|---|---|
| 7<br>FDIVLD | **Clock Divider Loaded**<br>0 FCLKDIV register has not been written<br>1 FCLKDIV register has been written since the last reset |
| 6–0<br>FDIV[6:0] | **Clock Divider Bits** — FDIV[6:0] must be set to effectively divide OSCCLK down to generate an internal Flash clock, FCLK, with a target frequency of 1 MHz for use by the Flash module to control timed events during program and erase algorithms. Table 25-9 shows recommended values for FDIV[6:0] based on OSCCLK frequency. Please refer to Section 25.4.1, “Flash Command Operations,” for more information. |

**CAUTION**

The FCLKDIV register should never be written while a Flash command is executing (CCIF=0). The FCLKDIV register is writable during the Flash reset sequence even though CCIF is clear.

**Table 25-9. FDIV vs OSCCLK Frequency**

| OSCCLK Frequency (MHz) | | FDIV[6:0] | OSCCLK Frequency (MHz) | | FDIV[6:0] | OSCCLK Frequency (MHz) | | FDIV[6:0] |
|---|---|---|---|---|---|---|---|---|
| MIN(1) | MAX(2) | | MIN1 | MAX2 | | MIN1 | MAX2 | |
| | | | 33.60 | 34.65 | 0x20 | 67.20 | 68.25 | 0x40 |
| 1.60 | 2.10 | 0x01 | 34.65 | 35.70 | 0x21 | 68.25 | 69.30 | 0x41 |
| 2.40 | 3.15 | 0x02 | 35.70 | 36.75 | 0x22 | 69.30 | 70.35 | 0x42 |
| 3.20 | 4.20 | 0x03 | 36.75 | 37.80 | 0x23 | 70.35 | 71.40 | 0x43 |
| 4.20 | 5.25 | 0x04 | 37.80 | 38.85 | 0x24 | 71.40 | 72.45 | 0x44 |
| 5.25 | 6.30 | 0x05 | 38.85 | 39.90 | 0x25 | 72.45 | 73.50 | 0x45 |
| 6.30 | 7.35 | 0x06 | 39.90 | 40.95 | 0x26 | 73.50 | 74.55 | 0x46 |
| 7.35 | 8.40 | 0x07 | 40.95 | 42.00 | 0x27 | 74.55 | 75.60 | 0x47 |
| 8.40 | 9.45 | 0x08 | 42.00 | 43.05 | 0x28 | 75.60 | 76.65 | 0x48 |
| 9.45 | 10.50 | 0x09 | 43.05 | 44.10 | 0x29 | 76.65 | 77.70 | 0x49 |
| 10.50 | 11.55 | 0x0A | 44.10 | 45.15 | 0x2A | 77.70 | 78.75 | 0x4A |
| 11.55 | 12.60 | 0x0B | 45.15 | 46.20 | 0x2B | 78.75 | 79.80 | 0x4B |
| 12.60 | 13.65 | 0x0C | 46.20 | 47.25 | 0x2C | 79.80 | 80.85 | 0x4C |
| 13.65 | 14.70 | 0x0D | 47.25 | 48.30 | 0x2D | 80.85 | 81.90 | 0x4D |
| 14.70 | 15.75 | 0x0E | 48.30 | 49.35 | 0x2E | 81.90 | 82.95 | 0x4E |
| 15.75 | 16.80 | 0x0F | 49.35 | 50.40 | 0x2F | 82.95 | 84.00 | 0x4F |
| 16.80 | 17.85 | 0x10 | 50.40 | 51.45 | 0x30 | 84.00 | 85.05 | 0x50 |
| 17.85 | 18.90 | 0x11 | 51.45 | 52.50 | 0x31 | 85.05 | 86.10 | 0x51 |
| 18.90 | 19.95 | 0x12 | 52.50 | 53.55 | 0x32 | 86.10 | 87.15 | 0x52 |
| 19.95 | 21.00 | 0x13 | 53.55 | 54.60 | 0x33 | 87.15 | 88.20 | 0x53 |
| 21.00 | 22.05 | 0x14 | 54.60 | 55.65 | 0x34 | 88.20 | 89.25 | 0x54 |
| 22.05 | 23.10 | 0x15 | 55.65 | 56.70 | 0x35 | 89.25 | 90.30 | 0x55 |
| 23.10 | 24.15 | 0x16 | 56.70 | 57.75 | 0x36 | 90.30 | 91.35 | 0x56 |
| 24.15 | 25.20 | 0x17 | 57.75 | 58.80 | 0x37 | 91.35 | 92.40 | 0x57 |
| 25.20 | 26.25 | 0x18 | 58.80 | 59.85 | 0x38 | 92.40 | 93.45 | 0x58 |
| 26.25 | 27.30 | 0x19 | 59.85 | 60.90 | 0x39 | 93.45 | 94.50 | 0x59 |
| 27.30 | 28.35 | 0x1A | 60.90 | 61.95 | 0x3A | 94.50 | 95.55 | 0x5A |
| 28.35 | 29.40 | 0x1B | 61.95 | 63.00 | 0x3B | 95.55 | 96.60 | 0x5B |
| 29.40 | 30.45 | 0x1C | 63.00 | 64.05 | 0x3C | 96.60 | 97.65 | 0x5C |
| 30.45 | 31.50 | 0x1D | 64.05 | 65.10 | 0x3D | 97.65 | 98.70 | 0x5D |
| 31.50 | 32.55 | 0x1E | 65.10 | 66.15 | 0x3E | 98.70 | 99.75 | 0x5E |
| 32.55 | 33.60 | 0x1F | 66.15 | 67.20 | 0x3F | 99.75 | 100.80 | 0x5F |

1. FDIV shown generates an FCLK frequency of >0.8 MHz

2. FDIV shown generates an FCLK frequency of 1.05 MHz

### 25.3.2.2 Flash Security Register (FSEC)

The FSEC register holds all bits associated with the security of the MCU and Flash module.

Offset Module Base + 0x0001

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | KEYEN[1:0] | | RNV[5:2] | | | | SEC[1:0] | |
| W | | | | | | | | |
| Reset | F | F | F | F | F | F | F | F |

= Unimplemented or Reserved

**Figure 25-6. Flash Security Register (FSEC)**

All bits in the FSEC register are readable but not writable.

During the reset sequence, the FSEC register is loaded with the contents of the Flash security byte in the Flash configuration field at global address 0x7F_FF0F located in P-Flash memory (see Table 25-3) as indicated by reset condition F in Figure 25-6. If a double bit fault is detected while reading the P-Flash phrase containing the Flash security byte during the reset sequence, all bits in the FSEC register will be set to leave the Flash module in a secured state with backdoor key access disabled.

**Table 25-10. FSEC Field Descriptions**

| Field | Description |
|---|---|
| 7–6<br>KEYEN[1:0] | **Backdoor Key Security Enable Bits** — The KEYEN[1:0] bits define the enabling of backdoor key access to the Flash module as shown in Table 25-11. |
| 5–2<br>RNV[5:2} | **Reserved Nonvolatile Bits** — The RNV bits should remain in the erased state for future enhancements. |
| 1–0<br>SEC[1:0] | **Flash Security Bits** — The SEC[1:0] bits define the security state of the MCU as shown in Table 25-12. If the Flash module is unsecured using backdoor key access, the SEC bits are forced to 10. |

**Table 25-11. Flash KEYEN States**

| KEYEN[1:0] | Status of Backdoor Key Access |
|---|---|
| 00 | DISABLED |
| 01 | DISABLED(1) |
| 10 | ENABLED |
| 11 | DISABLED |

1. Preferred KEYEN state to disable backdoor key access.

**Table 25-12. Flash Security States**

| SEC[1:0] | Status of Security |
|---|---|
| 00 | SECURED |
| 01 | SECURED(1) |
| 10 | UNSECURED |
| 11 | SECURED |

1. Preferred SEC state to set MCU to secured state.

The security function in the Flash module is described in Section 25.5.

### 25.3.2.3 Flash CCOB Index Register (FCCOBIX)

The FCCOBIX register is used to index the FCCOB register for Flash memory operations.

Offset Module Base + 0x0002

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | 0 | 0 | 0 | CCOBIX[2:0] | | |
| W | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented or Reserved

**Figure 25-7. FCCOB Index Register (FCCOBIX)**

CCOBIX bits are readable and writable while remaining bits read 0 and are not writable.

**Table 25-13. FCCOBIX Field Descriptions**

| Field | Description |
|---|---|
| 2–0 CCOBIX[1:0] | **Common Command Register Index**— The CCOBIX bits are used to select which word of the FCCOB register array is being read or written to. See Section 25.3.2.11, “Flash Common Command Object Register (FCCOB),” for more details. |

### 25.3.2.4 Flash ECCR Index Register (FECCRIX)

The FECCRIX register is used to index the FECCR register for ECC fault reporting.

Offset Module Base + 0x0003

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | 0 | 0 | 0 | ECCRIX[2:0] | | |
| W | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented or Reserved

**Figure 25-8. FECCR Index Register (FECCRIX)**

ECCRIX bits are readable and writable while remaining bits read 0 and are not writable.

**Table 25-14. FECCRIX Field Descriptions**

| Field | Description |
|---|---|
| 2-0<br>ECCRIX[2:0] | **ECC Error Register Index**— The ECCRIX bits are used to select which word of the FECCR register array is being read. See Section 25.3.2.13, “Flash ECC Error Results Register (FECCR),” for more details. |

### 25.3.2.5 Flash Configuration Register (FCNFG)

The FCNFG register enables the Flash command complete interrupt and forces ECC faults on Flash array read access from the CPU or XGATE.

Offset Module Base + 0x0004

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | CCIE | 0 | 0 | IGNSF | 0 | 0 | FDFD | FSFD |
| W | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented or Reserved

**Figure 25-9. Flash Configuration Register (FCNFG)**

CCIE, IGNSF, FDFD, and FSFD bits are readable and writable while remaining bits read 0 and are not writable.

**Table 25-15. FCNFG Field Descriptions**

| Field | Description |
|---|---|
| 7<br>CCIE | **Command Complete Interrupt Enable** — The CCIE bit controls interrupt generation when a Flash command has completed.<br>0 Command complete interrupt disabled<br>1 An interrupt will be requested whenever the CCIF flag in the FSTAT register is set (see Section 25.3.2.7) |
| 4<br>IGNSF | **Ignore Single Bit Fault** — The IGNSF controls single bit fault reporting in the FERSTAT register (see Section 25.3.2.8).<br>0 All single bit faults detected during array reads are reported<br>1 Single bit faults detected during array reads are not reported and the single bit fault interrupt will not be generated |

**Table 25-15. FCNFG Field Descriptions (continued)**

| Field | Description |
|---|---|
| 1<br>FDFD | **Force Double Bit Fault Detect** — The FDFD bit allows the user to simulate a double bit fault during Flash array read operations and check the associated interrupt routine. The FDFD bit is cleared by writing a 0 to FDFD. The FECCR registers will not be updated during the Flash array read operation with FDFD set unless an actual double bit fault is detected.<br>0 Flash array read operations will set the DFDIF flag in the FERSTAT register only if a double bit fault is detected<br>1 Any Flash array read operation will force the DFDIF flag in the FERSTAT register to be set (see Section 25.3.2.7) and an interrupt will be generated as long as the DFDIE interrupt enable in the FERCNFG register is set (see Section 25.3.2.6) |
| 0<br>FSFD | **Force Single Bit Fault Detect** — The FSFD bit allows the user to simulate a single bit fault during Flash array read operations and check the associated interrupt routine. The FSFD bit is cleared by writing a 0 to FSFD. The FECCR registers will not be updated during the Flash array read operation with FSFD set unless an actual single bit fault is detected.<br>0 Flash array read operations will set the SFDIF flag in the FERSTAT register only if a single bit fault is detected<br>1 Flash array read operation will force the SFDIF flag in the FERSTAT register to be set (see Section 25.3.2.7) and an interrupt will be generated as long as the SFDIE interrupt enable in the FERCNFG register is set (see Section 25.3.2.6) |

### 25.3.2.6 Flash Error Configuration Register (FERCNFG)

The FERCNFG register enables the Flash error interrupts for the FERSTAT flags.

Offset Module Base + 0x0005

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | ERSERIE | PGMERIE | 0 | EPVIOLIE | ERSVIE1 | ERSVIE0 | DFDIE | SFDIE |
| W | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented or Reserved

**Figure 25-10. Flash Error Configuration Register (FERCNFG)**

All assigned bits in the FERCNFG register are readable and writable.

**Table 25-16. FERCNFG Field Descriptions**

| Field | Description |
|---|---|
| 7<br>ERSERIE | **EEE Erase Error Interrupt Enable** — The ERSERIE bit controls interrupt generation when a failure is detected during an EEE erase operation.<br>0 ERSERIF interrupt disabled<br>1 An interrupt will be requested whenever the ERSERIF flag is set (see Section 25.3.2.8) |
| 6<br>PGMERIE | **EEE Program Error Interrupt Enable** — The PGMERIE bit controls interrupt generation when a failure is detected during an EEE program operation.<br>0 PGMERIF interrupt disabled<br>1 An interrupt will be requested whenever the PGMERIF flag is set (see Section 25.3.2.8) |
| 4<br>EPVIOLIE | **EEE Protection Violation Interrupt Enable** — The EPVIOLIE bit controls interrupt generation when a protection violation is detected during a write to the buffer RAM EEE partition.<br>0 EPVIOLIF interrupt disabled<br>1 An interrupt will be requested whenever the EPVIOLIF flag is set (see Section 25.3.2.8) |

**Table 25-16. FERCNFG Field Descriptions (continued)**

| Field | Description |
|---|---|
| 3<br>ERSVIE1 | **EEE Error Type 1 Interrupt Enable** — The ERSVIE1 bit controls interrupt generation when a change state error is detected during an EEE operation.<br>0 ERSVIF1 interrupt disabled<br>1 An interrupt will be requested whenever the ERSVIF1 flag is set (see Section 25.3.2.8) |
| 2<br>ERSVIE0 | **EEE Error Type 0 Interrupt Enable** — The ERSVIE0 bit controls interrupt generation when a sector format error is detected during an EEE operation.<br>0 ERSVIF0 interrupt disabled<br>1 An interrupt will be requested whenever the ERSVIF0 flag is set (see Section 25.3.2.8) |
| 1<br>DFDIE | **Double Bit Fault Detect Interrupt Enable** — The DFDIE bit controls interrupt generation when a double bit fault is detected during a Flash block read operation.<br>0 DFDIF interrupt disabled<br>1 An interrupt will be requested whenever the DFDIF flag is set (see Section 25.3.2.8) |
| 0<br>SFDIE | **Single Bit Fault Detect Interrupt Enable** — The SFDIE bit controls interrupt generation when a single bit fault is detected during a Flash block read operation.<br>0 SFDIF interrupt disabled whenever the SFDIF flag is set (see Section 25.3.2.8)<br>1 An interrupt will be requested whenever the SFDIF flag is set (see Section 25.3.2.8) |

### 25.3.2.7 Flash Status Register (FSTAT)

The FSTAT register reports the operational status of the Flash module.

Offset Module Base + 0x0006

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | CCIF | 0 | ACCERR | FPVIOL | MGBUSY | RSVD | MGSTAT[1:0] | |
| W | | | | | | | | |
| Reset | 1 | 0 | 0 | 0 | 0 | 0 | 0(1) | 0[1] |

= Unimplemented or Reserved

**Figure 25-11. Flash Status Register (FSTAT)**

1. Reset value can deviate from the value shown if a double bit fault is detected during the reset sequence (see Section 25.6).

CCIF, ACCERR, and FPVIOL bits are readable and writable, MGBUSY and MGSTAT bits are readable but not writable, while remaining bits read 0 and are not writable.

**Table 25-17. FSTAT Field Descriptions**

| Field | Description |
|---|---|
| 7<br>CCIF | **Command Complete Interrupt Flag** — The CCIF flag indicates that a Flash command has completed. The CCIF flag is cleared by writing a 1 to CCIF to launch a command and CCIF will stay low until command completion or command violation.<br>0 Flash command in progress<br>1 Flash command has completed |
| 5<br>ACCERR | **Flash Access Error Flag** — The ACCERR bit indicates an illegal access has occurred to the Flash memory caused by either a violation of the command write sequence (see Section 25.4.1.2) or issuing an illegal Flash command or when errors are encountered while initializing the EEE buffer ram during the reset sequence. While ACCERR is set, the CCIF flag cannot be cleared to launch a command. The ACCERR bit is cleared by writing a 1 to ACCERR. Writing a 0 to the ACCERR bit has no effect on ACCERR.<br>0 No access error detected<br>1 Access error detected |
| 4<br>FPVIOL | **Flash Protection Violation Flag** —The FPVIOL bit indicates an attempt was made to program or erase an address in a protected area of P-Flash memory during a command write sequence. The FPVIOL bit is cleared by writing a 1 to FPVIOL. Writing a 0 to the FPVIOL bit has no effect on FPVIOL. While FPVIOL is set, it is not possible to launch a command or start a command write sequence.<br>0 No protection violation detected<br>1 Protection violation detected |
| 3<br>MGBUSY | **Memory Controller Busy Flag** — The MGBUSY flag reflects the active state of the Memory Controller.<br>0 Memory Controller is idle<br>1 Memory Controller is busy executing a Flash command (CCIF = 0) or is handling internal EEE operations |
| 2<br>RSVD | **Reserved Bit** — This bit is reserved and always reads 0. |
| 1–0<br>MGSTAT[1:0] | **Memory Controller Command Completion Status Flag** — One or more MGSTAT flag bits are set if an error is detected during execution of a Flash command or during the Flash reset sequence. See Section 25.4.2, “Flash Command Description,” and Section 25.6, “Initialization” for details. |

### 25.3.2.8 Flash Error Status Register (FERSTAT)

The FERSTAT register reflects the error status of internal Flash operations.

Offset Module Base + 0x0007

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | ERSERIF | PGMERIF | 0 | EPVIOLIF | ERSVIF1 | ERSVIF0 | DFDIF | SFDIF |
| W | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented or Reserved

**Figure 25-12. Flash Error Status Register (FERSTAT)**

All flags in the FERSTAT register are readable and only writable to clear the flag.

**Table 25-18. FERSTAT Field Descriptions**

| Field | Description |
|---|---|
| 7<br>ERSERIF | **EEE Erase Error Interrupt Flag** — The setting of the ERSERIF flag occurs due to an error in a Flash erase command that resulted in the erase operation not being successful during EEE operations. The ERSERIF flag is cleared by writing a 1 to ERSERIF. Writing a 0 to the ERSERIF flag has no effect on ERSERIF. While ERSERIF is set, it is possible to write to the buffer RAM EEE partition but the data written will not be transferred to the D-Flash EEE partition.<br>0 Erase command successfully completed on the D-Flash EEE partition<br>1 Erase command failed on the D-Flash EEE partition |
| 6<br>PGMERIF | **EEE Program Error Interrupt Flag** — The setting of the PGMERIF flag occurs due to an error in a Flash program command that resulted in the program operation not being successful during EEE operations. The PGMERIF flag is cleared by writing a 1 to PGMERIF. Writing a 0 to the PGMERIF flag has no effect on PGMERIF. While PGMERIF is set, it is possible to write to the buffer RAM EEE partition but the data written will not be transferred to the D-Flash EEE partition.<br>0 Program command successfully completed on the D-Flash EEE partition<br>1 Program command failed on the D-Flash EEE partition |
| 4<br>EPVIOLIF | **EEE Protection Violation Interrupt Flag** —The setting of the EPVIOLIF flag indicates an attempt was made to write to a protected area of the buffer RAM EEE partition. The EPVIOLIF flag is cleared by writing a 1 to EPVIOLIF. Writing a 0 to the EPVIOLIF flag has no effect on EPVIOLIF. While EPVIOLIF is set, it is possible to write to the buffer RAM EEE partition as long as the address written to is not in a protected area.<br>0 No EEE protection violation<br>1 EEE protection violation detected |
| 3<br>ERSVIF1 | **EEE Error Interrupt 1 Flag** —The setting of the ERSVIF1 flag indicates that the memory controller was unable to change the state of a D-Flash EEE sector. The ERSVIF1 flag is cleared by writing a 1 to ERSVIF1. Writing a 0 to the ERSVIF1 flag has no effect on ERSVIF1. While ERSVIF1 is set, it is possible to write to the buffer RAM EEE partition but the data written will not be transferred to the D-Flash EEE partition.<br>0 No EEE sector state change error detected<br>1 EEE sector state change error detected |
| 2<br>ERSVIF0 | **EEE Error Interrupt 0 Flag** —The setting of the ERSVIF0 flag indicates that the memory controller was unable to format a D-Flash EEE sector for EEE use. The ERSVIF0 flag is cleared by writing a 1 to ERSVIF0. Writing a 0 to the ERSVIF0 flag has no effect on ERSVIF0. While ERSVIF0 is set, it is possible to write to the buffer RAM EEE partition but the data written will not be transferred to the D-Flash EEE partition.<br>0 No EEE sector format error detected<br>1 EEE sector format error detected |
| 1<br>DFDIF | **Double Bit Fault Detect Interrupt Flag** — The setting of the DFDIF flag indicates that a double bit fault was detected in the stored parity and data bits during a Flash array read operation or that a Flash array read operation was attempted on a Flash block that was under a Flash command operation. The DFDIF flag is cleared by writing a 1 to DFDIF. Writing a 0 to DFDIF has no effect on DFDIF.<br>0 No double bit fault detected<br>1 Double bit fault detected or an invalid Flash array read operation attempted |
| 0<br>SFDIF | **Single Bit Fault Detect Interrupt Flag** — With the IGNSF bit in the FCNFG register clear, the SFDIF flag indicates that a single bit fault was detected in the stored parity and data bits during a Flash array read operation or that a Flash array read operation was attempted on a Flash block that was under a Flash command operation. The SFDIF flag is cleared by writing a 1 to SFDIF. Writing a 0 to SFDIF has no effect on SFDIF.<br>0 No single bit fault detected<br>1 Single bit fault detected and corrected or an invalid Flash array read operation attempted |

### 25.3.2.9 P-Flash Protection Register (FPROT)

The FPROT register defines which P-Flash sectors are protected against program and erase operations.

Offset Module Base + 0x0008

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | FPOPEN | RNV6 | FPHDIS | FPHS[1:0] | | FPLDIS | FPLS[1:0] | |
| W | | | | | | | | |
| Reset | F | F | F | F | F | F | F | F |

= Unimplemented or Reserved

**Figure 25-13. Flash Protection Register (FPROT)**

The (unreserved) bits of the FPROT register are writable with the restriction that the size of the protected region can only be increased (see Section 25.3.2.9.1, “P-Flash Protection Restrictions,” and Table 25-23).

During the reset sequence, the FPROT register is loaded with the contents of the P-Flash protection byte in the Flash configuration field at global address 0x7F_FF0C located in P-Flash memory (see Table 25-3) as indicated by reset condition ‘F’ in Figure 25-13. To change the P-Flash protection that will be loaded during the reset sequence, the upper sector of the P-Flash memory must be unprotected, then the P-Flash protection byte must be reprogrammed. If a double bit fault is detected while reading the P-Flash phrase containing the P-Flash protection byte during the reset sequence, the FPOPEN bit will be cleared and remaining bits in the FPROT register will be set to leave the P-Flash memory fully protected.

Trying to alter data in any protected area in the P-Flash memory will result in a protection violation error and the FPVIOL bit will be set in the FSTAT register. The block erase of a P-Flash block is not possible if any of the P-Flash sectors contained in the same P-Flash block are protected.

**Table 25-19. FPROT Field Descriptions**

| Field | Description |
|---|---|
| 7<br>FPOPEN | **Flash Protection Operation Enable** — The FPOPEN bit determines the protection function for program or erase operations as shown in Table 25-20 for the P-Flash block.<br>0 When FPOPEN is clear, the FPHDIS and FPLDIS bits define unprotected address ranges as specified by the corresponding FPHS and FPLS bits<br>1 When FPOPEN is set, the FPHDIS and FPLDIS bits enable protection for the address range specified by the corresponding FPHS and FPLS bits |
| 6<br>RNV[6] | **Reserved Nonvolatile Bit** — The RNV bit should remain in the erased state for future enhancements. |
| 5<br>FPHDIS | **Flash Protection Higher Address Range Disable** — The FPHDIS bit determines whether there is a protected/unprotected area in a specific region of the P-Flash memory ending with global address 0x7F_FFFF.<br>0 Protection/Unprotection enabled<br>1 Protection/Unprotection disabled |
| 4–3<br>FPHS[1:0] | **Flash Protection Higher Address Size** — The FPHS bits determine the size of the protected/unprotected area in P-Flash memory as shown inTable 25-21. The FPHS bits can only be written to while the FPHDIS bit is set. |
| 2<br>FPLDIS | **Flash Protection Lower Address Range Disable** — The FPLDIS bit determines whether there is a protected/unprotected area in a specific region of the P-Flash memory beginning with global address 0x7F_8000.<br>0 Protection/Unprotection enabled<br>1 Protection/Unprotection disabled |
| 1–0<br>FPLS[1:0] | **Flash Protection Lower Address Size** — The FPLS bits determine the size of the protected/unprotected area in P-Flash memory as shown in Table 25-22. The FPLS bits can only be written to while the FPLDIS bit is set. |

**Table 25-20. P-Flash Protection Function**

| FPOPEN | FPHDIS | FPLDIS | Function[1] |
|---|---|---|---|
| 1 | 1 | 1 | No P-Flash Protection |
| 1 | 1 | 0 | Protected Low Range |
| 1 | 0 | 1 | Protected High Range |
| 1 | 0 | 0 | Protected High and Low Ranges |
| 0 | 1 | 1 | Full P-Flash Memory Protected |
| 0 | 1 | 0 | Unprotected Low Range |
| 0 | 0 | 1 | Unprotected High Range |
| 0 | 0 | 0 | Unprotected High and Low Ranges |

1. For range sizes, refer to Table 25-21 and Table 25-22.

**Table 25-21. P-Flash Protection Higher Address Range**

| FPHS[1:0] | Global Address Range | Protected Size |
|---|---|---|
| 00 | 0x7F_F800–0x7F_FFFF | 2 Kbytes |
| 01 | 0x7F_F000–0x7F_FFFF | 4 Kbytes |
| 10 | 0x7F_E000–0x7F_FFFF | 8 Kbytes |
| 11 | 0x7F_C000–0x7F_FFFF | 16 Kbytes |

**Table 25-22. P-Flash Protection Lower Address Range**

| FPLS[1:0] | Global Address Range | Protected Size |
|---|---|---|
| 00 | 0x7F_8000–0x7F_83FF | 1 Kbyte |
| 01 | 0x7F_8000–0x7F_87FF | 2 Kbytes |
| 10 | 0x7F_8000–0x7F_8FFF | 4 Kbytes |
| 11 | 0x7F_8000–0x7F_9FFF | 8 Kbytes |

All possible P-Flash protection scenarios are shown in Figure 25-14. Although the protection scheme is loaded from the Flash memory at global address 0x7F_FF0C during the reset sequence, it can be changed by the user. The P-Flash protection scheme can be used by applications requiring reprogramming in single chip mode while providing as much protection as possible if reprogramming is not required.

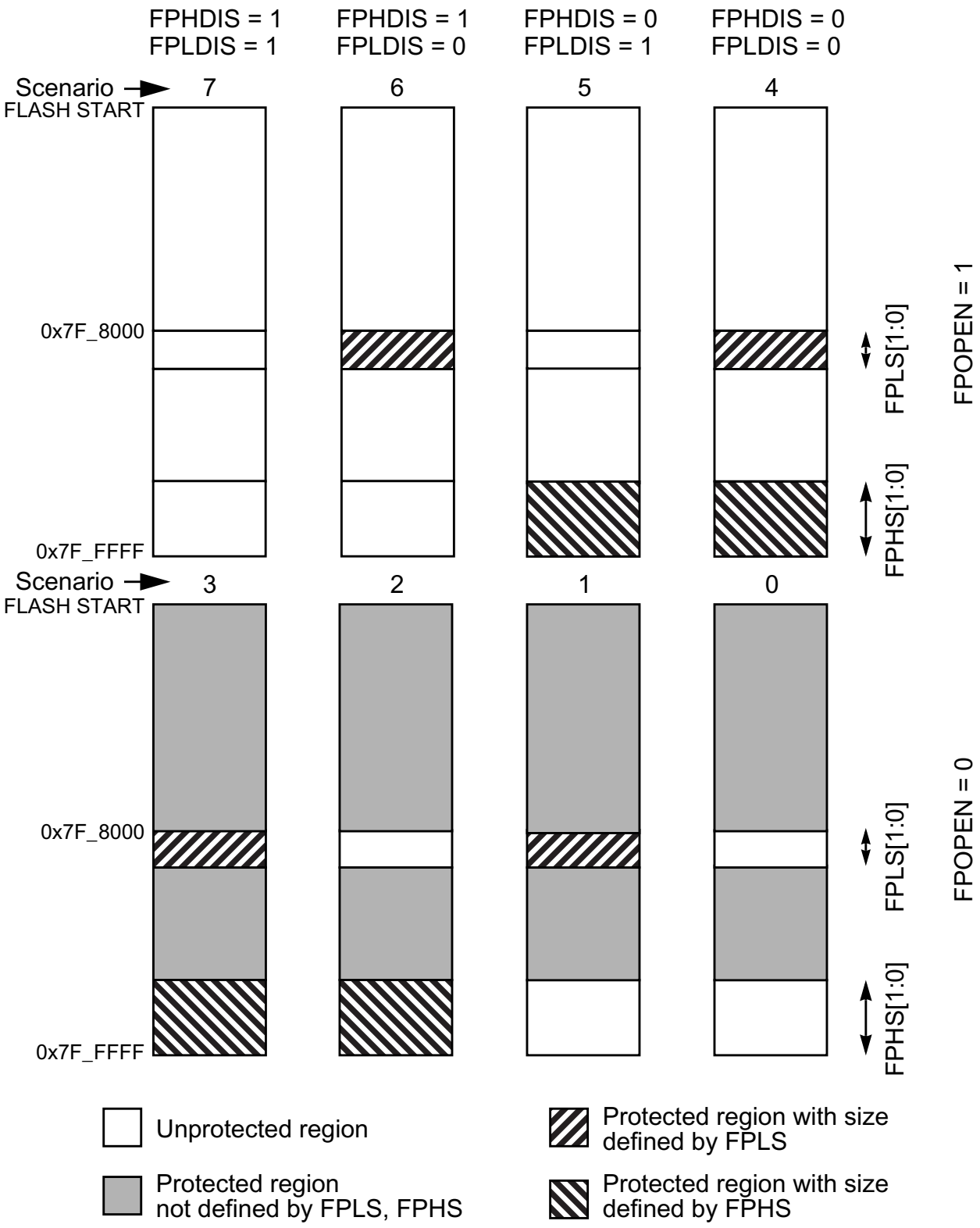



**Figure 25-14. P-Flash Protection Scenarios**

#### 25.3.2.9.1 P-Flash Protection Restrictions

The general guideline is that P-Flash protection can only be added and not removed. Table 25-23 specifies all valid transitions between P-Flash protection scenarios. Any attempt to write an invalid scenario to the FPROT register will be ignored. The contents of the FPROT register reflect the active protection scenario. See the FPHS and FPLS bit descriptions for additional restrictions.

**Table 25-23. P-Flash Protection Scenario Transitions**

| From Protection Scenario | To Protection Scenario[1] | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| 0 | X | X | X | X | | | | |
| 1 | | X | | X | | | | |
| 2 | | | X | X | | | | |
| 3 | | | | X | | | | |
| 4 | | | | X | X | | | |
| 5 | | | X | X | X | X | | |
| 6 | | X | | X | X | | X | |
| 7 | X | X | X | X | X | X | X | X |

1. Allowed transitions marked with X, see Figure 25-14 for a definition of the scenarios.

### 25.3.2.10 EEE Protection Register (EPROT)

The EPROT register defines which buffer RAM EEE partition areas are protected against writes.

Offset Module Base + 0x0009

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | EPOPEN | RNV[6:4] | | | EPDIS | EPS[2:0] | | |
| W | | | | | | | | |
| Reset | F | F | F | F | F | F | F | F |

= Unimplemented or Reserved

**Figure 25-15. EEE Protection Register (EPROT)**

All bits in the EPROT register are readable and writable except for RNV[6:4] which are only readable. The EPOPEN and EPDIS bits can only be written to the protected state. The EPS bits can be written anytime until the EPDIS bit is cleared. If the EPOPEN bit is cleared, the state of the EPDIS and EPS bits is irrelevant.

During the reset sequence, the EPROT register is loaded from the EEE protection byte in the Flash configuration field at global address 0x7F_FF0D located in P-Flash memory (see Table 25-3) as indicated by reset condition F in Figure 25-15. To change the EEE protection that will be loaded during the reset sequence, the P-Flash sector containing the EEE protection byte must be unprotected, then the EEE protection byte must be programmed. If a double bit fault is detected while reading the P-Flash phrase

containing the EEE protection byte during the reset sequence, the EPOPEN bit will be cleared and remaining bits in the EPROT register will be set to leave the buffer RAM EEE partition fully protected.

Trying to write data to any protected area in the buffer RAM EEE partition will result in a protection violation error and the EPVIOLIF flag will be set in the FERSTAT register. Trying to write data to any protected area in the buffer RAM partitioned for user access will not be prevented and the EPVIOLIF flag in the FERSTAT register will not set.

**Table 25-24. EPROT Field Descriptions**

| Field | Description |
|---|---|
| 7<br>EPOPEN | **Enables writes to the Buffer RAM partitioned for EEE**<br>0 The entire buffer RAM EEE partition is protected from writes<br>1 Unprotected buffer RAM EEE partition areas are enabled for writes |
| 6–4<br>RNV[6:4] | **Reserved Nonvolatile Bits** — The RNV bits should remain in the erased state for future enhancements |
| 3<br>EPDIS | **Buffer RAM Protection Address Range Disable** — The EPDIS bit determines whether there is a protected area in a specific region of the buffer RAM EEE partition.<br>0 Protection enabled<br>1 Protection disabled |
| 2–0<br>EPS[2:0] | **Buffer RAM Protection Size** — The EPS[2:0] bits determine the size of the protected area in the buffer RAM EEE partition as shown inTable 25-21. The EPS bits can only be written to while the EPDIS bit is set. |

**Table 25-25. Buffer RAM EEE Partition Protection Address Range**

| EPS[2:0] | Global Address Range | Protected Size |
|---|---|---|
| 000 | 0x13_FFC0 – 0x13_FFFF | 64 bytes |
| 001 | 0x13_FF80 – 0x13_FFFF | 128 bytes |
| 010 | 0x13_FF40 – 0x13_FFFF | 192 bytes |
| 011 | 0x13_FF00 – 0x13_FFFF | 256 bytes |
| 100 | 0x13_FEC0 – 0x13_FFFF | 320 bytes |
| 101 | 0x13_FE80 – 0x13_FFFF | 384 bytes |
| 110 | 0x13_FE40 – 0x13_FFFF | 448 bytes |
| 111 | 0x13_FE00 – 0x13_FFFF | 512 bytes |

### 25.3.2.11 Flash Common Command Object Register (FCCOB)

The FCCOB is an array of six words addressed via the CCOBIX index found in the FCCOBIX register. Byte wide reads and writes are allowed to the FCCOB register.

Offset Module Base + 0x000A

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | CCOB[15:8] | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 25-16. Flash Common Command Object High Register (FCCOBHI)**

Offset Module Base + 0x000B

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | CCOB[7:0] | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 25-17. Flash Common Command Object Low Register (FCCOBLO)**

### 25.3.2.11.1 FCCOB - NVM Command Mode

NVM command mode uses the indexed FCCOB register to provide a command code and its relevant parameters to the Memory Controller. The user first sets up all required FCCOB fields and then initiates the command's execution by writing a 1 to the CCIF bit in the FSTAT register (a 1 written by the user clears the CCIF command completion flag to 0). When the user clears the CCIF bit in the FSTAT register all FCCOB parameter fields are locked and cannot be changed by the user until the command completes (as evidenced by the Memory Controller returning CCIF to 1). Some commands return information to the FCCOB register array.

The generic format for the FCCOB parameter fields in NVM command mode is shown in Table 25-26. The return values are available for reading after the CCIF flag in the FSTAT register has been returned to 1 by the Memory Controller. Writes to the unimplemented parameter fields (CCOBIX = 110 and CCOBIX = 111) are ignored with reads from these fields returning 0x0000.

Table 25-26 shows the generic Flash command format. The high byte of the first word in the CCOB array contains the command code, followed by the parameters for this specific Flash command. For details on the FCCOB settings required by each command, see the Flash command descriptions in Section 25.4.2.

**Table 25-26. FCCOB - NVM Command Mode (Typical Usage)**

| CCOBIX[2:0] | Byte | FCCOB Parameter Fields (NVM Command Mode) |
|---|---|---|
| 000 | HI | FCMD[7:0] defining Flash command |
| | LO | 0, Global address [22:16] |
| 001 | HI | Global address [15:8] |
| | LO | Global address [7:0] |
| 010 | HI | Data 0 [15:8] |
| | LO | Data 0 [7:0] |

**Table 25-26. FCCOB - NVM Command Mode (Typical Usage)**

| CCOBIX[2:0] | Byte | FCCOB Parameter Fields (NVM Command Mode) |
|---|---|---|
| 011 | HI | Data 1 [15:8] |
| | LO | Data 1 [7:0] |
| 100 | HI | Data 2 [15:8] |
| | LO | Data 2 [7:0] |
| 101 | HI | Data 3 [15:8] |
| | LO | Data 3 [7:0] |

### 25.3.2.12 EEE Tag Counter Register (ETAG)

The ETAG register contains the number of outstanding words in the buffer RAM EEE partition that need to be programmed into the D-Flash EEE partition. The ETAG register is decremented prior to the related tagged word being programmed into the D-Flash EEE partition. All tagged words have been programmed into the D-Flash EEE partition once all bits in the ETAG register read 0 and the MGBUSY flag in the FSTAT register reads 0.

Offset Module Base + 0x000C

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | ETAG[15:8] | | | | | | | |
| W | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented or Reserved

**Figure 25-18. EEE Tag Counter High Register (ETAGHI)**

Offset Module Base + 0x000D

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | ETAG[7:0] | | | | | | | |
| W | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented or Reserved

**Figure 25-19. EEE Tag Counter Low Register (ETAGLO)**

All ETAG bits are readable but not writable and are cleared by the Memory Controller.

### 25.3.2.13 Flash ECC Error Results Register (FECCR)

The FECCR registers contain the result of a detected ECC fault for both single bit and double bit faults. The FECCR register provides access to several ECC related fields as defined by the ECCRIX index bits in the FECCRIX register (see Section 25.3.2.4). Once ECC fault information has been stored, no other

fault information will be recorded until the specific ECC fault flag has been cleared. In the event of simultaneous ECC faults, the priority for fault recording is:

1. Double bit fault over single bit fault
2. CPU over XGATE

Offset Module Base + 0x000E

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | ECCR[15:8] | | | | | | | |
| W | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented or Reserved

**Figure 25-20. Flash ECC Error Results High Register (FECCRHI)**

Offset Module Base + 0x000F

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | ECCR[7:0] | | | | | | | |
| W | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented or Reserved

**Figure 25-21. Flash ECC Error Results Low Register (FECCRLO)**

All FECCR bits are readable but not writable.

**Table 25-27. FECCR Index Settings**

| ECCRIX[2:0] | FECCR Register Content | | |
|---|---|---|---|
| | **Bits [15:8]** | **Bit[7]** | **Bits[6:0]** |
| 000 | Parity bits read from Flash block | CPU or XGATE source identity | Global address [22:16] |
| 001 | Global address [15:0] | | |
| 010 | Data 0 [15:0] | | |
| 011 | Data 1 [15:0] (P-Flash only) | | |
| 100 | Data 2 [15:0] (P-Flash only) | | |
| 101 | Data 3 [15:0] (P-Flash only) | | |
| 110 | Not used, returns 0x0000 when read | | |
| 111 | Not used, returns 0x0000 when read | | |

**Table 25-28. FECCR Index=000 Bit Descriptions**

| Field | Description |
|---|---|
| 15:8<br>PAR[7:0] | **ECC Parity Bits** — Contains the 8 parity bits from the 72 bit wide P-Flash data word or the 6 parity bits, allocated to PAR[5:0], from the 22 bit wide D-Flash word with PAR[7:6]=00. |
| 7<br>XBUS01 | **Bus Source Identifier** — The XBUS01 bit determines whether the ECC error was caused by a read access from the CPU or XGATE.<br>0 ECC Error happened on the CPU access<br>1 ECC Error happened on the XGATE access |
| 6–0<br>GADDR[22:16] | **Global Address** — The GADDR[22:16] field contains the upper seven bits of the global address having caused the error. |

The P-Flash word addressed by ECCRIX = 001 contains the lower 16 bits of the global address. The following four words addressed by ECCRIX = 010 to 101 contain the 64-bit wide data phrase. The four data words and the parity byte are the uncorrected data read from the P-Flash block.

The D-Flash word addressed by ECCRIX = 001 contains the lower 16 bits of the global address. The uncorrected 16-bit data word is addressed by ECCRIX = 010.

### 25.3.2.14 Flash Option Register (FOPT)

The FOPT register is the Flash option register.

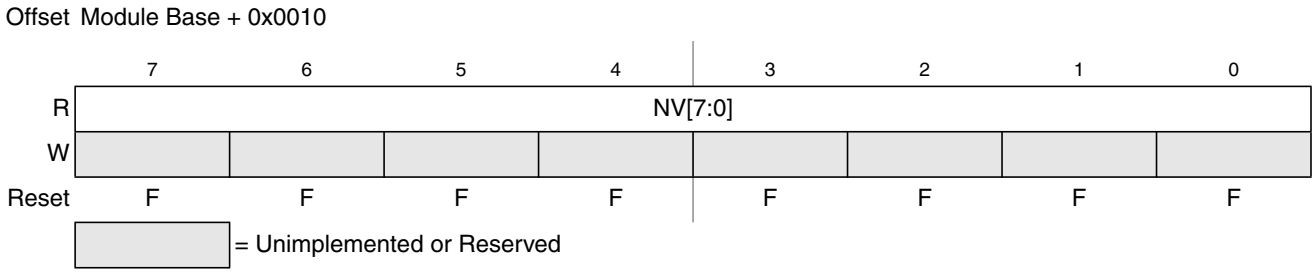



**Figure 25-22. Flash Option Register (FOPT)**

All bits in the FOPT register are readable but are not writable.

During the reset sequence, the FOPT register is loaded from the Flash nonvolatile byte in the Flash configuration field at global address 0x7F_FF0E located in P-Flash memory (see Table 25-3) as indicated by reset condition F in Figure 25-22. If a double bit fault is detected while reading the P-Flash phrase containing the Flash nonvolatile byte during the reset sequence, all bits in the FOPT register will be set.

**Table 25-29. FOPT Field Descriptions**

| Field | Description |
|---|---|
| 7–0<br>NV[7:0] | **Nonvolatile Bits** — The NV[7:0] bits are available as nonvolatile bits. Refer to the device user guide for proper use of the NV bits. |

### 25.3.2.15 Flash Reserved0 Register (FRSV0)

This Flash register is reserved for factory testing.

Offset Module Base + 0x0011

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| W | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented or Reserved

**Figure 25-23. Flash Reserved0 Register (FRSV0)**

All bits in the FRSV0 register read 0 and are not writable.

### 25.3.2.16 Flash Reserved1 Register (FRSV1)

This Flash register is reserved for factory testing.

Offset Module Base + 0x0012

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| W | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented or Reserved

**Figure 25-24. Flash Reserved1 Register (FRSV1)**

All bits in the FRSV1 register read 0 and are not writable.

### 25.3.2.17 Flash Reserved2 Register (FRSV2)

This Flash register is reserved for factory testing.

Offset Module Base + 0x0013

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| W | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented or Reserved

**Figure 25-25. Flash Reserved2 Register (FRSV2)**

All bits in the FRSV2 register read 0 and are not writable.

## 25.4 Functional Description

### 25.4.1 Flash Command Operations

Flash command operations are used to modify Flash memory contents or configure module resources for EEE operation.

The next sections describe:

- How to write the FCLKDIV register that is used to generate a time base (FCLK) derived from OSCCLK for Flash program and erase command operations
- The command write sequence used to set Flash command parameters and launch execution
- Valid Flash commands available for execution

#### 25.4.1.1 Writing the FCLKDIV Register

Prior to issuing any Flash program or erase command after a reset, the user is required to write the FCLKDIV register to divide OSCCLK down to a target FCLK of 1 MHz. Table 25-9 shows recommended values for the FDIV field based on OSCCLK frequency.

**NOTE**

Programming or erasing the Flash memory cannot be performed if the bus clock runs at less than 1 MHz. Setting FDIV too high can destroy the Flash memory due to overstress. Setting FDIV too low can result in incomplete programming or erasure of the Flash memory cells.

When the FCLKDIV register is written, the FDIVLD bit is set automatically. If the FDIVLD bit is 0, the FCLKDIV register has not been written since the last reset. If the FCLKDIV register has not been written, any Flash program or erase command loaded during a command write sequence will not execute and the ACCERR bit in the FSTAT register will set.

#### 25.4.1.2 Command Write Sequence

The Memory Controller will launch all valid Flash commands entered using a command write sequence.

Before launching a command, the ACCERR and FPVIOL bits in the FSTAT register must be clear (see Section 25.3.2.7) and the CCIF flag should be tested to determine the status of the current command write sequence. If CCIF is 0, the previous command write sequence is still active, a new command write sequence cannot be started, and all writes to the FCCOB register are ignored.

**CAUTION**

Writes to any Flash register must be avoided while a Flash command is active (CCIF=0) to prevent corruption of Flash register contents and Memory Controller behavior.

#### 25.4.1.2.1 Define FCCOB Contents

The FCCOB parameter fields must be loaded with all required parameters for the Flash command being executed. Access to the FCCOB parameter fields is controlled via the CCOBIX bits in the FCCOBIX register (see Section 25.3.2.3).

The contents of the FCCOB parameter fields are transferred to the Memory Controller when the user clears the CCIF command completion flag in the FSTAT register (writing 1 clears the CCIF to 0). The CCIF flag will remain clear until the Flash command has completed. Upon completion, the Memory Controller will return CCIF to 1 and the FCCOB register will be used to communicate any results. The flow for a generic command write sequence is shown in Figure 25-26.

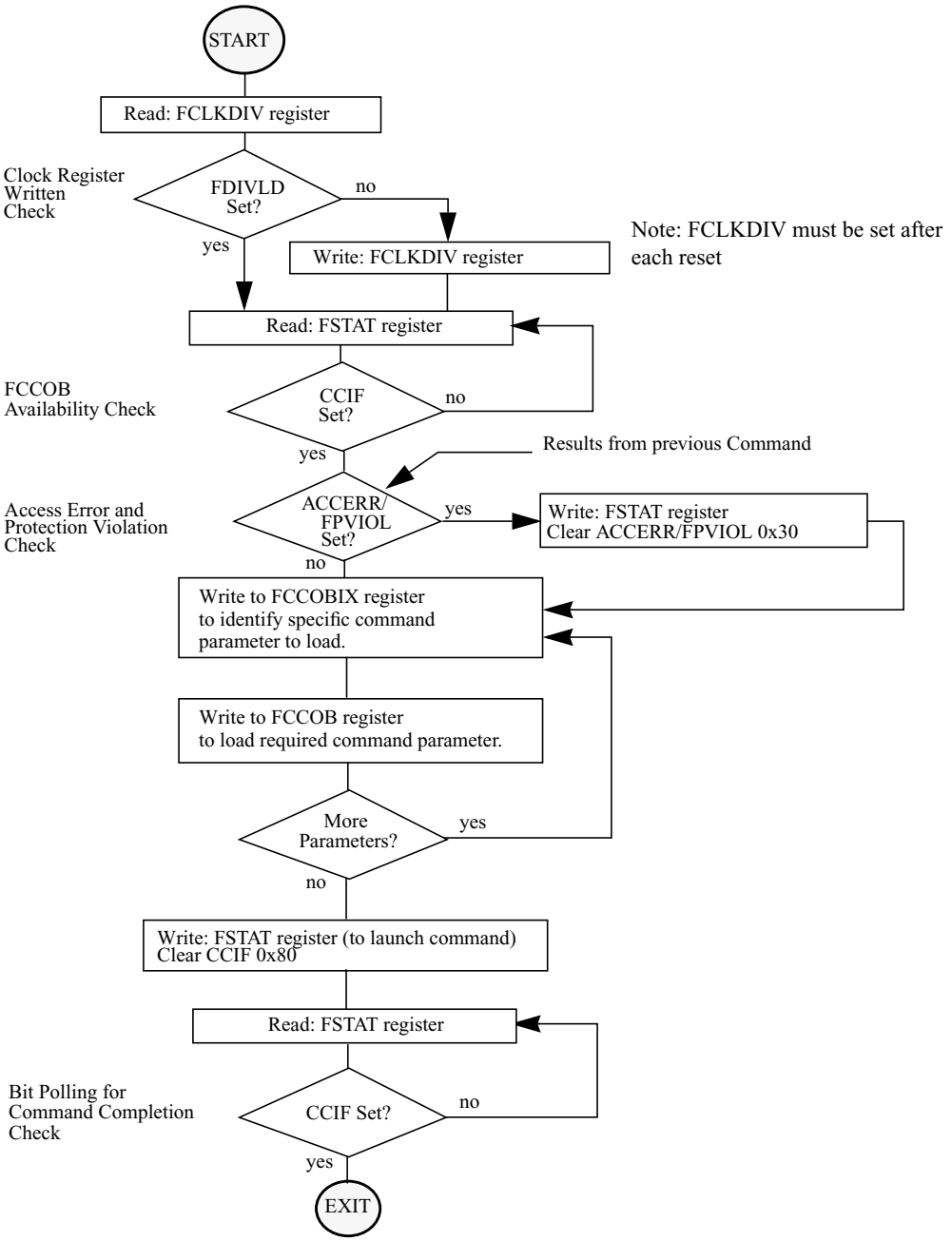



**Figure 25-26. Generic Flash Command Write Sequence Flowchart**

### 25.4.1.3 Valid Flash Module Commands

**Table 25-30. Flash Commands by Mode**

| FCMD | Command | Unsecured | | | | Secured | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | NS (1) | NX (2) | SS(3) | ST(4) | NS (5) | NX (6) | SS(7) | ST(8) |
| 0x01 | Erase Verify All Blocks | * | * | * | * | * | * | * | * |
| 0x02 | Erase Verify Block | * | * | * | * | * | * | * | * |
| 0x03 | Erase Verify P-Flash Section | * | * | * | * | * | | | |
| 0x04 | Read Once | * | * | * | * | * | | | |
| 0x05 | Load Data Field | * | * | * | * | * | | | |
| 0x06 | Program P-Flash | * | * | * | * | * | | | |
| 0x07 | Program Once | * | * | * | * | * | | | |
| 0x08 | Erase All Blocks | | | * | * | | | * | * |
| 0x09 | Erase P-Flash Block | * | * | * | * | * | | | |
| 0x0A | Erase P-Flash Sector | * | * | * | * | * | | | |
| 0x0B | Unsecure Flash | | | * | * | | | * | * |
| 0x0C | Verify Backdoor Access Key | * | | | | * | | | |
| 0x0D | Set User Margin Level | * | * | * | * | * | | | |
| 0x0E | Set Field Margin Level | | | * | * | | | | |
| 0x0F | Full Partition D-Flash | | | * | * | | | | |
| 0x10 | Erase Verify D-Flash Section | * | * | * | * | * | | | |
| 0x11 | Program D-Flash | * | * | * | * | * | | | |
| 0x12 | Erase D-Flash Sector | * | * | * | * | * | | | |
| 0x13 | Enable EEPROM Emulation | * | * | * | * | * | * | * | * |
| 0x14 | Disable EEPROM Emulation | * | * | * | * | * | * | * | * |
| 0x15 | EEPROM Emulation Query | * | * | * | * | * | * | * | * |
| 0x20 | Partition D-Flash | * | * | * | * | * | * | * | * |

1. Unsecured Normal Single Chip mode.
2. Unsecured Normal Expanded mode.
3. Unsecured Special Single Chip mode.
4. Unsecured Special Mode.
5. Secured Normal Single Chip mode.
6. Secured Normal Expanded mode.
7. Secured Special Single Chip mode.
8. Secured Special Mode.

#### 25.4.1.4 P-Flash Commands

Table 25-31 summarizes the valid P-Flash commands along with the effects of the commands on the P-Flash block and other resources within the Flash module.

**Table 25-31. P-Flash Commands**

| FCMD | Command | Function on P-Flash Memory |
|---|---|---|
| 0x01 | Erase Verify All Blocks | Verify that all P-Flash (and D-Flash) blocks are erased. |
| 0x02 | Erase Verify Block | Verify that a P-Flash block is erased. |
| 0x03 | Erase Verify P-Flash Section | Verify that a given number of words starting at the address provided are erased. |
| 0x04 | Read Once | Read a dedicated 64 byte field in the nonvolatile information register in P-Flash block 0 that was previously programmed using the Program Once command. |
| 0x05 | Load Data Field | Load data for simultaneous multiple P-Flash block operations. |
| 0x06 | Program P-Flash | Program a phrase in a P-Flash block and any previously loaded phrases for any other P-Flash block (see Load Data Field command). |
| 0x07 | Program Once | Program a dedicated 64 byte field in the nonvolatile information register in P-Flash block 0 that is allowed to be programmed only once. |
| 0x08 | Erase All Blocks | Erase all P-Flash (and D-Flash) blocks.<br>An erase of all Flash blocks is only possible when the FPLDIS, FPHDIS, and FPOPEN bits in the FPROT register and the EPDIS and EPOPEN bits in the EPROT register are set prior to launching the command. |
| 0x09 | Erase P-Flash Block | Erase a single P-Flash block.<br>An erase of the full P-Flash block is only possible when FPLDIS, FPHDIS and FPOPEN bits in the FPROT register are set prior to launching the command. |
| 0x0A | Erase P-Flash Sector | Erase all bytes in a P-Flash sector. |
| 0x0B | Unsecure Flash | Supports a method of releasing MCU security by erasing all P-Flash (and D-Flash) blocks and verifying that all P-Flash (and D-Flash) blocks are erased. |
| 0x0C | Verify Backdoor Access Key | Supports a method of releasing MCU security by verifying a set of security keys. |
| 0x0D | Set User Margin Level | Specifies a user margin read level for all P-Flash blocks. |
| 0x0E | Set Field Margin Level | Specifies a field margin read level for all P-Flash blocks (special modes only). |

#### 25.4.1.5 D-Flash and EEE Commands

Table 25-32 summarizes the valid D-Flash and EEE commands along with the effects of the commands on the D-Flash block and EEE operation.

**Table 25-32. D-Flash Commands**

| FCMD | Command | Function on D-Flash Memory |
|---|---|---|
| 0x01 | Erase Verify All Blocks | Verify that all D-Flash (and P-Flash) blocks are erased. |
| 0x02 | Erase Verify Block | Verify that the D-Flash block is erased. |

**Table 25-32. D-Flash Commands**

| FCMD | Command | Function on D-Flash Memory |
|---|---|---|
| 0x08 | Erase All Blocks | Erase all D-Flash (and P-Flash) blocks.<br>An erase of all Flash blocks is only possible when the FPLDIS, FPHDIS, and FPOPEN bits in the FPROT register and the EPDIS and EPOPEN bits in the EPROT register are set prior to launching the command. |
| 0x0B | Unsecure Flash | Supports a method of releasing MCU security by erasing all D-Flash (and P-Flash) blocks and verifying that all D-Flash (and P-Flash) blocks are erased. |
| 0x0D | Set User Margin Level | Specifies a user margin read level for the D-Flash block. |
| 0x0E | Set Field Margin Level | Specifies a field margin read level for the D-Flash block (special modes only). |
| 0x0F | Full Partition D-Flash | Erase the D-Flash block and partition an area of the D-Flash block for user access. |
| 0x10 | Erase Verify D-Flash Section | Verify that a given number of words starting at the address provided are erased. |
| 0x11 | Program D-Flash | Program up to four words in the D-Flash block. |
| 0x12 | Erase D-Flash Sector | Erase all bytes in a sector of the D-Flash block. |
| 0x13 | Enable EEPROM Emulation | Enable EEPROM emulation where writes to the buffer RAM EEE partition will be copied to the D-Flash EEE partition. |
| 0x14 | Disable EEPROM Emulation | Suspend all current erase and program activity related to EEPROM emulation but leave current EEE tags set. |
| 0x15 | EEPROM Emulation Query | Returns EEE partition and status variables. |
| 0x20 | Partition D-Flash | Partition an area of the D-Flash block for user access. |

## 25.4.2 Flash Command Description

This section provides details of all available Flash commands launched by a command write sequence. The ACCERR bit in the FSTAT register will be set during the command write sequence if any of the following illegal steps are performed, causing the command not to be processed by the Memory Controller:

- Starting any command write sequence that programs or erases Flash memory before initializing the FCLKDIV register
- Writing an invalid command as part of the command write sequence
- For additional possible errors, refer to the error handling table provided for each command

If a Flash block is read during execution of an algorithm (CCIF = 0) on that same block, the read operation will return invalid data. If the SFDIF or DFDIF flags were not previously set when the invalid read operation occurred, both the SFDIF and DFDIF flags will be set and the FECCR registers will be loaded with the global address used in the invalid read operation with the data and parity fields set to all 0.

If the ACCERR or FPVIOL bits are set in the FSTAT register, the user must clear these bits before starting any command write sequence (see Section 25.3.2.7).

**CAUTION**

A Flash word or phrase must be in the erased state before being programmed. Cumulative programming of bits within a Flash word or phrase is not allowed.

### 25.4.2.1 Erase Verify All Blocks Command

The Erase Verify All Blocks command will verify that all P-Flash and D-Flash blocks have been erased.

**Table 25-33. Erase Verify All Blocks Command FCCOB Requirements**

| CCOBIX[2:0] | FCCOB Parameters | |
|---|---|---|
| 000 | 0x01 | Not required |

Upon clearing CCIF to launch the Erase Verify All Blocks command, the Memory Controller will verify that the entire Flash memory space is erased. The CCIF flag will set after the Erase Verify All Blocks operation has completed.

**Table 25-34. Erase Verify All Blocks Command Error Handling**

| Register | Error Bit | Error Condition |
|---|---|---|
| FSTAT | ACCERR | Set if CCOBIX[2:0] != 000 at command launch |
| | | Set if a Load Data Field command sequence is currently active |
| | FPVIOL | None |
| | MGSTAT1 | Set if any errors have been encountered during the read |
| | MGSTAT0 | Set if any non-correctable errors have been encountered during the read |
| FERSTAT | EPVIOLIF | None |

### 25.4.2.2 Erase Verify Block Command

The Erase Verify Block command allows the user to verify that an entire P-Flash or D-Flash block has been erased. The FCCOB upper global address bits determine which block must be verified.

**Table 25-35. Erase Verify Block Command FCCOB Requirements**

| CCOBIX[2:0] | FCCOB Parameters | |
|---|---|---|
| 000 | 0x02 | Global address [22:16] of the Flash block to be verified. |

Upon clearing CCIF to launch the Erase Verify Block command, the Memory Controller will verify that the selected P-Flash or D-Flash block is erased. The CCIF flag will set after the Erase Verify Block operation has completed.

**Table 25-36. Erase Verify Block Command Error Handling**

| Register | Error Bit | Error Condition |
|---|---|---|
| FSTAT | ACCERR | Set if CCOBIX[2:0] != 000 at command launch |
| | | Set if a Load Data Field command sequence is currently active |
| | | Set if an invalid global address [22:16] is supplied |
| | FPVIOL | None |
| | MGSTAT1 | Set if any errors have been encountered during the read |
| | MGSTAT0 | Set if any non-correctable errors have been encountered during the read |
| FERSTAT | EPVIOLIF | None |

### 25.4.2.3 Erase Verify P-Flash Section Command

The Erase Verify P-Flash Section command will verify that a section of code in the P-Flash memory is erased. The Erase Verify P-Flash Section command defines the starting point of the code to be verified and the number of phrases. The section to be verified cannot cross a 128 Kbyte boundary in the P-Flash memory space.

**Table 25-37. Erase Verify P-Flash Section Command FCCOB Requirements**

| CCOBIX[2:0] | FCCOB Parameters | |
|---|---|---|
| 000 | 0x03 | Global address [22:16] of a P-Flash block |
| 001 | Global address [15:0] of the first phrase to be verified | |
| 010 | Number of phrases to be verified | |

Upon clearing CCIF to launch the Erase Verify P-Flash Section command, the Memory Controller will verify the selected section of Flash memory is erased. The CCIF flag will set after the Erase Verify P-Flash Section operation has completed.

**Table 25-38. Erase Verify P-Flash Section Command Error Handling**

| Register | Error Bit | Error Condition |
|---|---|---|
| FSTAT | ACCERR | Set if CCOBIX[2:0] != 010 at command launch |
| | | Set if a Load Data Field command sequence is currently active |
| | | Set if command not available in current mode (see Table 25-30) |
| | | Set if an invalid global address [22:0] is supplied |
| | | Set if a misaligned phrase address is supplied (global address [2:0] != 000) |
| | | Set if the requested section crosses a 128 Kbyte boundary |
| | FPVIOL | None |
| | MGSTAT1 | Set if any errors have been encountered during the read |
| | MGSTAT0 | Set if any non-correctable errors have been encountered during the read |

**Table 25-38. Erase Verify P-Flash Section Command Error Handling**

| Register | Error Bit | Error Condition |
|---|---|---|
| FERSTAT | EPVIOLIF | None |

### 25.4.2.4 Read Once Command

The Read Once command provides read access to a reserved 64 byte field (8 phrases) located in the nonvolatile information register of P-Flash block 0. The Read Once field is programmed using the Program Once command described in Section 25.4.2.7. The Read Once command must not be executed from the Flash block containing the Program Once reserved field to avoid code runaway.

**Table 25-39. Read Once Command FCCOB Requirements**

| CCOBIX[2:0] | FCCOB Parameters | |
|---|---|---|
| 000 | 0x04 | Not Required |
| 001 | Read Once phrase index (0x0000 - 0x0007) | |
| 010 | Read Once word 0 value | |
| 011 | Read Once word 1 value | |
| 100 | Read Once word 2 value | |
| 101 | Read Once word 3 value | |

Upon clearing CCIF to launch the Read Once command, a Read Once phrase is fetched and stored in the FCCOB indexed register. The CCIF flag will set after the Read Once operation has completed. Valid phrase index values for the Read Once command range from 0x0000 to 0x0007. During execution of the Read Once command, any attempt to read addresses within P-Flash block 0 will return invalid data.

**Table 25-40. Read Once Command Error Handling**

| Register | Error Bit | Error Condition |
|---|---|---|
| FSTAT | ACCERR | Set if CCOBIX[2:0] != 001 at command launch |
| | | Set if a Load Data Field command sequence is currently active |
| | | Set if command not available in current mode (see Table 25-30) |
| | | Set if an invalid phrase index is supplied |
| | FPVIOL | None |
| | MGSTAT1 | Set if any errors have been encountered during the read |
| | MGSTAT0 | Set if any non-correctable errors have been encountered during the read |
| FERSTAT | EPVIOLIF | None |

### 25.4.2.5 Load Data Field Command

The Load Data Field command is executed to provide FCCOB parameters for multiple P-Flash blocks for a future simultaneous program operation in the P-Flash memory space.

**Table 25-41. Load Data Field Command FCCOB Requirements**

| CCOBIX[2:0] | FCCOB Parameters | |
|---|---|---|
| 000 | 0x05 | Global address [22:16] to identify P-Flash block |
| 001 | Global address [15:0] of phrase location to be programmed(1) | |
| 010 | Word 0 | |
| 011 | Word 1 | |
| 100 | Word 2 | |
| 101 | Word 3 | |

1. Global address [2:0] must be 000

Upon clearing CCIF to launch the Load Data Field command, the FCCOB registers will be transferred to the Memory Controller and be programmed in the block specified at the global address given with a future Program P-Flash command launched on a P-Flash block. The CCIF flag will set after the Load Data Field operation has completed. Note that once a Load Data Field command sequence has been initiated, the Load Data Field command sequence will be cancelled if any command other than Load Data Field or the future Program P-Flash is launched. Similarly, if an error occurs after launching a Load Data Field or Program P-Flash command, the associated Load Data Field command sequence will be cancelled.

**Table 25-42. Load Data Field Command Error Handling**

| Register | Error Bit | Error Condition |
|---|---|---|
| FSTAT | ACCERR | Set if CCOBIX[2:0] != 101 at command launch |
| | | Set if command not available in current mode (see Table 25-30) |
| | | Set if an invalid global address [22:0] is supplied |
| | | Set if a misaligned phrase address is supplied (global address [2:0] != 000) |
| | | Set if a Load Data Field command sequence is currently active and the selected block has previously been selected in the same command sequence |
| | | Set if a Load Data Field command sequence is currently active and global address [16:0] does not match that previously supplied in the same command sequence |
| | FPVIOL | Set if the global address [22:0] points to a protected area |
| | MGSTAT1 | None |
| | MGSTAT0 | None |
| FERSTAT | EPVIOLIF | None |

### 25.4.2.6 Program P-Flash Command

The Program P-Flash operation will program a previously erased phrase in the P-Flash memory using an embedded algorithm.

**CAUTION**

A P-Flash phrase must be in the erased state before being programmed. Cumulative programming of bits within a Flash phrase is not allowed.

**Table 25-43. Program P-Flash Command FCCOB Requirements**

| CCOBIX[2:0] | FCCOB Parameters | |
|---|---|---|
| 000 | 0x06 | Global address [22:16] to identify P-Flash block |
| 001 | Global address [15:0] of phrase location to be programmed(1) | |
| 010 | Word 0 program value | |
| 011 | Word 1 program value | |
| 100 | Word 2 program value | |
| 101 | Word 3 program value | |

1. Global address [2:0] must be 000

Upon clearing CCIF to launch the Program P-Flash command, the Memory Controller will program the data words to the supplied global address and will then proceed to verify the data words read back as expected. The CCIF flag will set after the Program P-Flash operation has completed.

**Table 25-44. Program P-Flash Command Error Handling**

| Register | Error Bit | Error Condition |
|---|---|---|
| FSTAT | ACCERR | Set if CCOBIX[2:0] != 101 at command launch |
| | | Set if command not available in current mode (see Table 25-30) |
| | | Set if an invalid global address [22:0] is supplied |
| | | Set if a misaligned phrase address is supplied (global address [2:0] != 000) |
| | | Set if a Load Data Field command sequence is currently active and the selected block has previously been selected in the same command sequence |
| | | Set if a Load Data Field command sequence is currently active and global address [17:0] does not match that previously supplied in the same command sequence |
| | FPVIOL | Set if the global address [22:0] points to a protected area |
| | MGSTAT1 | Set if any errors have been encountered during the verify operation |
| | MGSTAT0 | Set if any non-correctable errors have been encountered during the verify operation |
| FERSTAT | EPVIOLIF | None |

### 25.4.2.7 Program Once Command

The Program Once command restricts programming to a reserved 64 byte field (8 phrases) in the nonvolatile information register located in P-Flash block 0. The Program Once reserved field can be read using the Read Once command as described in Section 25.4.2.4. The Program Once command must only be issued once since the nonvolatile information register in P-Flash block 0 cannot be erased. The Program

Once command must not be executed from the Flash block containing the Program Once reserved field to avoid code runaway.

**Table 25-45. Program Once Command FCCOB Requirements**

| CCOBIX[2:0] | FCCOB Parameters | |
|---|---|---|
| 000 | 0x07 | Not Required |
| 001 | Program Once phrase index (0x0000 - 0x0007) | |
| 010 | Program Once word 0 value | |
| 011 | Program Once word 1 value | |
| 100 | Program Once word 2 value | |
| 101 | Program Once word 3 value | |

Upon clearing CCIF to launch the Program Once command, the Memory Controller first verifies that the selected phrase is erased. If erased, then the selected phrase will be programmed and then verified with read back. The CCIF flag will remain clear, setting only after the Program Once operation has completed.

The reserved nonvolatile information register accessed by the Program Once command cannot be erased and any attempt to program one of these phrases a second time will not be allowed. Valid phrase index values for the Program Once command range from 0x0000 to 0x0007. During execution of the Program Once command, any attempt to read addresses within P-Flash block 0 will return invalid data.

**Table 25-46. Program Once Command Error Handling**

| Register | Error Bit | Error Condition |
|---|---|---|
| FSTAT | ACCERR | Set if CCOBIX[2:0] != 101 at command launch |
| | | Set if a Load Data Field command sequence is currently active |
| | | Set if command not available in current mode (see Table 25-30) |
| | | Set if an invalid phrase index is supplied |
| | | Set if the requested phrase has already been programmed(1) |
| | FPVIOL | None |
| | MGSTAT1 | Set if any errors have been encountered during the verify operation |
| | MGSTAT0 | Set if any non-correctable errors have been encountered during the verify operation |
| FERSTAT | EPVIOLIF | None |

1. If a Program Once phrase is initially programmed to 0xFFFF_FFFF_FFFF_FFFF, the Program Once command will be allowed to execute again on that same phrase.

### 25.4.2.8 Erase All Blocks Command

The Erase All Blocks operation will erase the entire P-Flash and D-Flash memory space including the EEE nonvolatile information register.

**Table 25-47. Erase All Blocks Command FCCOB Requirements**

| CCOBIX[2:0] | FCCOB Parameters | |
|---|---|---|
| 000 | 0x08 | Not required |

Upon clearing CCIF to launch the Erase All Blocks command, the Memory Controller will erase the entire Flash memory space and verify that it is erased. If the Memory Controller verifies that the entire Flash memory space was properly erased, security will be released. During the execution of this command (CCIF=0) the user must not write to any Flash module register. The CCIF flag will set after the Erase All Blocks operation has completed.

**Table 25-48. Erase All Blocks Command Error Handling**

| Register | Error Bit | Error Condition |
|---|---|---|
| FSTAT | ACCERR | Set if CCOBIX[2:0] != 000 at command launch |
| | | Set if a Load Data Field command sequence is currently active |
| | | Set if command not available in current mode (see Table 25-30) |
| | FPVIOL | Set if any area of the P-Flash memory is protected |
| | MGSTAT1 | Set if any errors have been encountered during the verify operation |
| | MGSTAT0 | Set if any non-correctable errors have been encountered during the verify operation |
| FERSTAT | EPVIOLIF | Set if any area of the buffer RAM EEE partition is protected |

### 25.4.2.9 Erase P-Flash Block Command

The Erase P-Flash Block operation will erase all addresses in a P-Flash block.

**Table 25-49. Erase P-Flash Block Command FCCOB Requirements**

| CCOBIX[2:0] | FCCOB Parameters | |
|---|---|---|
| 000 | 0x09 | Global address [22:16] to identify P-Flash block |
| 001 | Global address [15:0] in P-Flash block to be erased | |

Upon clearing CCIF to launch the Erase P-Flash Block command, the Memory Controller will erase the selected P-Flash block and verify that it is erased. The CCIF flag will set after the Erase P-Flash Block operation has completed.

**Table 25-50. Erase P-Flash Block Command Error Handling**

| Register | Error Bit | Error Condition |
|---|---|---|
| FSTAT | ACCERR | Set if CCOBIX[2:0] != 001 at command launch |
| | | Set if a Load Data Field command sequence is currently active |
| | | Set if command not available in current mode (see Table 25-30) |
| | | Set if an invalid global address [22:16] is supplied |
| | FPVIOL | Set if an area of the selected P-Flash block is protected |
| | MGSTAT1 | Set if any errors have been encountered during the verify operation |
| | MGSTAT0 | Set if any non-correctable errors have been encountered during the verify operation |
| FERSTAT | EPVIOLIF | None |

### 25.4.2.10 Erase P-Flash Sector Command

The Erase P-Flash Sector operation will erase all addresses in a P-Flash sector.

**Table 25-51. Erase P-Flash Sector Command FCCOB Requirements**

| CCOBIX[2:0] | FCCOB Parameters | |
|---|---|---|
| 000 | 0x0A | Global address [22:16] to identify P-Flash block to be erased |
| 001 | Global address [15:0] anywhere within the sector to be erased. Refer to Section 25.1.2.1 for the P-Flash sector size. | |

Upon clearing CCIF to launch the Erase P-Flash Sector command, the Memory Controller will erase the selected Flash sector and then verify that it is erased. The CCIF flag will be set after the Erase P-Flash Sector operation has completed.

**Table 25-52. Erase P-Flash Sector Command Error Handling**

| Register | Error Bit | Error Condition |
|---|---|---|
| FSTAT | ACCERR | Set if CCOBIX[2:0] != 001 at command launch |
| | | Set if a Load Data Field command sequence is currently active |
| | | Set if command not available in current mode (see Table 25-30) |
| | | Set if an invalid global address [22:16] is supplied |
| | | Set if a misaligned phrase address is supplied (global address [2:0] != 000) |
| | FPVIOL | Set if the selected P-Flash sector is protected |
| | MGSTAT1 | Set if any errors have been encountered during the verify operation |
| | MGSTAT0 | Set if any non-correctable errors have been encountered during the verify operation |
| FERSTAT | EPVIOLIF | None |

### 25.4.2.11 Unsecure Flash Command

The Unsecure Flash command will erase the entire P-Flash and D-Flash memory space and, if the erase is successful, will release security.

**Table 25-53. Unsecure Flash Command FCCOB Requirements**

| CCOBIX[2:0] | FCCOB Parameters | |
|---|---|---|
| 000 | 0x0B | Not required |

Upon clearing CCIF to launch the Unsecure Flash command, the Memory Controller will erase the entire P-Flash and D-Flash memory space and verify that it is erased. If the Memory Controller verifies that the entire Flash memory space was properly erased, security will be released. If the erase verify is not successful, the Unsecure Flash operation sets MGSTAT1 and terminates without changing the security state. During the execution of this command (CCIF=0) the user must not write to any Flash module register. The CCIF flag is set after the Unsecure Flash operation has completed.

**Table 25-54. Unsecure Flash Command Error Handling**

| Register | Error Bit | Error Condition |
|---|---|---|
| FSTAT | ACCERR | Set if CCOBIX[2:0] != 000 at command launch |
| | | Set if a Load Data Field command sequence is currently active |
| | | Set if command not available in current mode (see Table 25-30) |
| | FPVIOL | Set if any area of the P-Flash memory is protected |
| | MGSTAT1 | Set if any errors have been encountered during the verify operation |
| | MGSTAT0 | Set if any non-correctable errors have been encountered during the verify operation |
| FERSTAT | EPVIOLIF | Set if any area of the buffer RAM EEE partition is protected |

### 25.4.2.12 Verify Backdoor Access Key Command

The Verify Backdoor Access Key command will only execute if it is enabled by the KEYEN bits in the FSEC register (see Table 25-11). The Verify Backdoor Access Key command releases security if user-supplied keys match those stored in the Flash security bytes of the Flash configuration field (see Table 25-3). The Verify Backdoor Access Key command must not be executed from the Flash block containing the backdoor comparison key to avoid code runaway.

**Table 25-55. Verify Backdoor Access Key Command FCCOB Requirements**

| CCOBIX[2:0] | FCCOB Parameters | |
|---|---|---|
| 000 | 0x0C | Not required |
| 001 | Key 0 | |
| 010 | Key 1 | |
| 011 | Key 2 | |
| 100 | Key 3 | |

Upon clearing CCIF to launch the Verify Backdoor Access Key command, the Memory Controller will check the FSEC KEYEN bits to verify that this command is enabled. If not enabled, the Memory Controller sets the ACCERR bit in the FSTAT register and terminates. If the command is enabled, the Memory Controller compares the key provided in FCCOB to the backdoor comparison key in the Flash configuration field with Key 0 compared to 0x7F_FF00, etc. If the backdoor keys match, security will be released. If the backdoor keys do not match, security is not released and all future attempts to execute the Verify Backdoor Access Key command are aborted (set ACCERR) until a reset occurs. The CCIF flag is set after the Verify Backdoor Access Key operation has completed.

**Table 25-56. Verify Backdoor Access Key Command Error Handling**

| Register | Error Bit | Error Condition |
|---|---|---|
| FSTAT | ACCERR | Set if CCOBIX[2:0] != 100 at command launch |
| | | Set if a Load Data Field command sequence is currently active |
| | | Set if an incorrect backdoor key is supplied |
| | | Set if backdoor key access has not been enabled (KEYEN[1:0] != 10, see Section 25.3.2.2) |
| | | Set if the backdoor key has mismatched since the last reset |
| | FPVIOL | None |
| | MGSTAT1 | None |
| | MGSTAT0 | None |
| FERSTAT | EPVIOLIF | None |

### 25.4.2.13 Set User Margin Level Command

The Set User Margin Level command causes the Memory Controller to set the margin level for future read operations of a specific P-Flash or D-Flash block.

**Table 25-57. Set User Margin Level Command FCCOB Requirements**

| CCOBIX[2:0] | FCCOB Parameters | |
|---|---|---|
| 000 | 0x0D | Global address [22:16] to identify the Flash block |
| 001 | Margin level setting | |

Upon clearing CCIF to launch the Set User Margin Level command, the Memory Controller will set the user margin level for the targeted block and then set the CCIF flag.

Valid margin level settings for the Set User Margin Level command are defined in Table 25-58.

**Table 25-58. Valid Set User Margin Level Settings**

| CCOB (CCOBIX=001) | Level Description |
|---|---|
| 0x0000 | Return to Normal Level |
| 0x0001 | User Margin-1 Level(1) |

**Table 25-58. Valid Set User Margin Level Settings**

| CCOB (CCOBIX=001) | Level Description |
|---|---|
| 0x0002 | User Margin-0 Level(2) |

1. Read margin to the erased state
2. Read margin to the programmed state

**Table 25-59. Set User Margin Level Command Error Handling**

| Register | Error Bit | Error Condition |
|---|---|---|
| FSTAT | ACCERR | Set if CCOBIX[2:0] != 001 at command launch |
| | | Set if a Load Data Field command sequence is currently active |
| | | Set if command not available in current mode (see Table 25-30) |
| | | Set if an invalid global address [22:16] is supplied |
| | | Set if an invalid margin level setting is supplied |
| | FPVIOL | None |
| | MGSTAT1 | None |
| | MGSTAT0 | None |
| FERSTAT | EPVIOLIF | None |

**NOTE**

User margin levels can be used to check that Flash memory contents have adequate margin for normal level read operations. If unexpected results are encountered when checking Flash memory contents at user margin levels, a potential loss of information has been detected.

### 25.4.2.14 Set Field Margin Level Command

The Set Field Margin Level command, valid in special modes only, causes the Memory Controller to set the margin level specified for future read operations of a specific P-Flash or D-Flash block.

**Table 25-60. Set Field Margin Level Command FCCOB Requirements**

| CCOBIX[2:0] | FCCOB Parameters | |
|---|---|---|
| 000 | 0x0E | Global address [22:16] to identify the Flash block |
| 001 | Margin level setting | |

Upon clearing CCIF to launch the Set Field Margin Level command, the Memory Controller will set the field margin level for the targeted block and then set the CCIF flag.

Valid margin level settings for the Set Field Margin Level command are defined in Table 25-61.

**Table 25-61. Valid Set Field Margin Level Settings**

| CCOB (CCOBIX=001) | Level Description |
|---|---|
| 0x0000 | Return to Normal Level |
| 0x0001 | User Margin-1 Level(1) |
| 0x0002 | User Margin-0 Level(2) |
| 0x0003 | Field Margin-1 Level[1] |
| 0x0004 | Field Margin-0 Level[2] |

1. Read margin to the erased state
2. Read margin to the programmed state

**Table 25-62. Set Field Margin Level Command Error Handling**

| Register | Error Bit | Error Condition |
|---|---|---|
| FSTAT | ACCERR | Set if CCOBIX[2:0] != 001 at command launch |
| | | Set if a Load Data Field command sequence is currently active |
| | | Set if command not available in current mode (see Table 25-30) |
| | | Set if an invalid global address [22:16] is supplied |
| | | Set if an invalid margin level setting is supplied |
| | FPVIOL | None |
| | MGSTAT1 | None |
| | MGSTAT0 | None |
| FERSTAT | EPVIOLIF | None |

**CAUTION**

Field margin levels must only be used during verify of the initial factory programming.

**NOTE**

Field margin levels can be used to check that Flash memory contents have adequate margin for data retention at the normal level setting. If unexpected results are encountered when checking Flash memory contents at field margin levels, the Flash memory contents should be erased and reprogrammed.

### 25.4.2.15 Full Partition D-Flash Command

The Full Partition D-Flash command allows the user to allocate sectors within the D-Flash block for applications and a partition within the buffer RAM for EEPROM access. The D-Flash block consists of 128 sectors with 256 bytes per sector.

**Table 25-63. Full Partition D-Flash Command FCCOB Requirements**

| CCOBIX[2:0] | FCCOB Parameters | |
|---|---|---|
| 000 | 0x0F | Not required |
| 001 | Number of 256 byte sectors for the D-Flash user partition (DFPART) | |
| 010 | Number of 256 byte sectors for buffer RAM EEE partition (ERPART) | |

Upon clearing CCIF to launch the Full Partition D-Flash command, the following actions are taken to define a partition within the D-Flash block for direct access (DFPART) and a partition within the buffer RAM for EEE use (ERPART):

- Validate the DFPART and ERPART values provided:
  - DFPART <= 128 (maximum number of 256 byte sectors in D-Flash block)
  - ERPART <= 16 (maximum number of 256 byte sectors in buffer RAM)
  - If ERPART > 0, 128 - DFPART >= 12 (minimum number of 256 byte sectors in the D-Flash block required to support EEE)
  - If ERPART > 0, ((128-DFPART)/ERPART) >= 8 (minimum ratio of D-Flash EEE space to buffer RAM EEE space to support EEE)
- Erase the D-Flash block and the EEE nonvolatile information register
- Program DFPART to the EEE nonvolatile information register at global address 0x12_0000 (see Table 25-7)
- Program a duplicate DFPART to the EEE nonvolatile information register at global address 0x12_0002 (see Table 25-7)
- Program ERPART to the EEE nonvolatile information register at global address 0x12_0004 (see Table 25-7)
- Program a duplicate ERPART to the EEE nonvolatile information register at global address 0x12_0006 (see Table 25-7)

The D-Flash user partition will start at global address 0x10_0000. The buffer RAM EEE partition will end at global address 0x13_FFFF. After the Full Partition D-Flash operation has completed, the CCIF flag will set.

Running the Full Partition D-Flash command a second time will result in the previous partition values and the entire D-Flash memory being erased. The data value written corresponds to the number of 256 byte sectors allocated for either direct D-Flash access (DFPART) or buffer RAM EEE access (ERPART).

**Table 25-64. Full Partition D-Flash Command Error Handling**

| Register | Error Bit | Error Condition |
|---|---|---|
| FSTAT | ACCERR | Set if CCOBIX[2:0] != 010 at command launch |
| | | Set if a Load Data Field command sequence is currently active |
| | | Set if command not available in current mode (see Table 25-30) |
| | | Set if an invalid DFPART or ERPART selection is supplied |
| | FPVIOL | None |
| | MGSTAT1 | Set if any errors have been encountered during the read |
| | MGSTAT0 | Set if any non-correctable errors have been encountered during the read |
| FERSTAT | EPVIOLIF | None |

### 25.4.2.16 Erase Verify D-Flash Section Command

The Erase Verify D-Flash Section command will verify that a section of code in the D-Flash user partition is erased. The Erase Verify D-Flash Section command defines the starting point of the data to be verified and the number of words.

**Table 25-65. Erase Verify D-Flash Section Command FCCOB Requirements**

| CCOBIX[2:0] | FCCOB Parameters | |
|---|---|---|
| 000 | 0x10 | Global address [22:16] to identify the D-Flash block |
| 001 | Global address [15:0] of the first word to be verified | |
| 010 | Number of words to be verified | |

Upon clearing CCIF to launch the Erase Verify D-Flash Section command, the Memory Controller will verify the selected section of D-Flash memory is erased. The CCIF flag will set after the Erase Verify D-Flash Section operation has completed.

**Table 25-66. Erase Verify D-Flash Section Command Error Handling**

| Register | Error Bit | Error Condition |
|---|---|---|
| FSTAT | ACCERR | Set if CCOBIX[2:0] != 010 at command launch |
| | | Set if a Load Data Field command sequence is currently active |
| | | Set if command not available in current mode (see Table 25-30) |
| | | Set if an invalid global address [22:0] is supplied |
| | | Set if a misaligned word address is supplied (global address [0] != 0) |
| | | Set if the global address [22:0] points to an area of the D-Flash EEE partition |
| | | Set if the requested section breaches the end of the D-Flash block or goes into the D-Flash EEE partition |
| | FPVIOL | None |
| | MGSTAT1 | Set if any errors have been encountered during the read |
| | MGSTAT0 | Set if any non-correctable errors have been encountered during the read |
| FERSTAT | EPVIOLIF | None |

### 25.4.2.17 Program D-Flash Command

The Program D-Flash operation programs one to four previously erased words in the D-Flash user partition. The Program D-Flash operation will confirm that the targeted location(s) were successfully programmed upon completion.

**CAUTION**

A Flash word must be in the erased state before being programmed. Cumulative programming of bits within a Flash word is not allowed.

**Table 25-67. Program D-Flash Command FCCOB Requirements**

| CCOBIX[2:0] | FCCOB Parameters | |
|---|---|---|
| 000 | 0x11 | Global address [22:16] to identify the D-Flash block |
| 001 | Global address [15:0] of word to be programmed | |
| 010 | Word 0 program value | |
| 011 | Word 1 program value, if desired | |
| 100 | Word 2 program value, if desired | |
| 101 | Word 3 program value, if desired | |

Upon clearing CCIF to launch the Program D-Flash command, the user-supplied words will be transferred to the Memory Controller and be programmed. The CCOBIX index value at Program D-Flash command launch determines how many words will be programmed in the D-Flash block. No protection checks are made in the Program D-Flash operation on the D-Flash block, only access error checks. The CCIF flag is set when the operation has completed.

**Table 25-68. Program D-Flash Command Error Handling**

| Register | Error Bit | Error Condition |
|---|---|---|
| FSTAT | ACCERR | Set if CCOBIX[2:0] < 010 at command launch |
| | | Set if CCOBIX[2:0] > 101 at command launch |
| | | Set if a Load Data Field command sequence is currently active |
| | | Set if command not available in current mode (see Table 25-30) |
| | | Set if an invalid global address [22:0] is supplied |
| | | Set if a misaligned word address is supplied (global address [0] != 0) |
| | | Set if the global address [22:0] points to an area in the D-Flash EEE partition |
| | | Set if the requested group of words breaches the end of the D-Flash block or goes into the D-Flash EEE partition |
| | FPVIOL | None |
| | MGSTAT1 | Set if any errors have been encountered during the verify operation |
| | MGSTAT0 | Set if any non-correctable errors have been encountered during the verify operation |
| FERSTAT | EPVIOLIF | None |

### 25.4.2.18 Erase D-Flash Sector Command

The Erase D-Flash Sector operation will erase all addresses in a sector of the D-Flash user partition.

**Table 25-69. Erase D-Flash Sector Command FCCOB Requirements**

| CCOBIX[2:0] | FCCOB Parameters | |
|---|---|---|
| 000 | 0x12 | Global address [22:16] to identify D-Flash block |
| 001 | Global address [15:0] anywhere within the sector to be erased. See Section 25.1.2.2 for D-Flash sector size. | |

Upon clearing CCIF to launch the Erase D-Flash Sector command, the Memory Controller will erase the selected Flash sector and verify that it is erased. The CCIF flag will set after the Erase D-Flash Sector operation has completed.

**Table 25-70. Erase D-Flash Sector Command Error Handling**

| Register | Error Bit | Error Condition |
|---|---|---|
| FSTAT | ACCERR | Set if CCOBIX[2:0] != 001 at command launch |
| | | Set if a Load Data Field command sequence is currently active |
| | | Set if command not available in current mode (see Table 25-30) |
| | | Set if an invalid global address [22:0] is supplied |
| | | Set if a misaligned word address is supplied (global address [0] != 0) |
| | | Set if the global address [22:0] points to the D-Flash EEE partition |
| | FPVIOL | None |
| | MGSTAT1 | Set if any errors have been encountered during the verify operation |
| | MGSTAT0 | Set if any non-correctable errors have been encountered during the verify operation |
| FERSTAT | EPVIOLIF | None |

### 25.4.2.19 Enable EEPROM Emulation Command

The Enable EEPROM Emulation command causes the Memory Controller to enable EEE activity. EEE activity is disabled after any reset.

**Table 25-71. Enable EEPROM Emulation Command FCCOB Requirements**

| CCOBIX[2:0] | FCCOB Parameters | |
|---|---|---|
| 000 | 0x13 | Not required |

Upon clearing CCIF to launch the Enable EEPROM Emulation command, the CCIF flag will set after the Memory Controller enables EEE operations using the contents of the EEE tag RAM and tag counter. The Full Partition D-Flash or the Partition D-Flash command must be run prior to launching the Enable EEPROM Emulation command.

**Table 25-72. Enable EEPROM Emulation Command Error Handling**

| Register | Error Bit | Error Condition |
|---|---|---|
| FSTAT | ACCERR | Set if CCOBIX[2:0] != 000 at command launch |
| | | Set if a Load Data Field command sequence is currently active |
| | | Set if Full Partition D-Flash or Partition D-Flash command not previously run |
| | FPVIOL | None |
| | MGSTAT1 | None |
| | MGSTAT0 | None |
| FERSTAT | EPVIOLIF | None |

### 25.4.2.20 Disable EEPROM Emulation Command

The Disable EEPROM Emulation command causes the Memory Controller to suspend current EEE activity.

**Table 25-73. Disable EEPROM Emulation Command FCCOB Requirements**

| CCOBIX[2:0] | FCCOB Parameters | |
|---|---|---|
| 000 | 0x14 | Not required |

Upon clearing CCIF to launch the Disable EEPROM Emulation command, the Memory Controller will halt EEE operations at the next convenient point without clearing the EEE tag RAM or tag counter before setting the CCIF flag.

**Table 25-74. Disable EEPROM Emulation Command Error Handling**

| Register | Error Bit | Error Condition |
|---|---|---|
| FSTAT | ACCERR | Set if CCOBIX[2:0] != 000 at command launch |
| | | Set if a Load Data Field command sequence is currently active |
| | | Set if Full Partition D-Flash or Partition D-Flash command not previously run |
| | FPVIOL | None |
| | MGSTAT1 | None |
| | MGSTAT0 | None |
| FERSTAT | EPVIOLIF | None |

### 25.4.2.21 EEPROM Emulation Query Command

The EEPROM Emulation Query command returns EEE partition and status variables.

**Table 25-75. EEPROM Emulation Query Command FCCOB Requirements**

| CCOBIX[2:0] | FCCOB Parameters | |
|---|---|---|
| 000 | 0x15 | Not required |
| 001 | Return DFPART | |
| 010 | Return ERPART | |
| 011 | Return ECOUNT(1) | |
| 100 | Return Dead Sector Count | Return Ready Sector Count |

1. Indicates sector erase count

Upon clearing CCIF to launch the EEPROM Emulation Query command, the CCIF flag will set after the EEE partition and status variables are stored in the FCCOBIX register.If the Emulation Query command is executed prior to partitioning (Partition D-Flash Command Section 25.4.2.15), the following reset values are returned: DFPART = 0x_FFFF, ERPART = 0x_FFFF, ECOUNT = 0x_FFFF, Dead Sector Count = 0x_00, Ready Sector Count = 0x_00.

**Table 25-76. EEPROM Emulation Query Command Error Handling**

| Register | Error Bit | Error Condition |
|---|---|---|
| FSTAT | ACCERR | Set if CCOBIX[2:0] != 000 at command launch |
| | | Set if a Load Data Field command sequence is currently active |
| | | Set if command not available in current mode (see Table 25-30) |
| | FPVIOL | None |
| | MGSTAT1 | None |
| | MGSTAT0 | None |
| FERSTAT | EPVIOLIF | None |

### 25.4.2.22 Partition D-Flash Command

The Partition D-Flash command allows the user to allocate sectors within the D-Flash block for applications and a partition within the buffer RAM for EEPROM access. The D-Flash block consists of 64 sectors with 256 bytes per sector. The Erase All Blocks command must be run prior to launching the Partition D-Flash command.

**Table 25-77. Partition D-Flash Command FCCOB Requirements**

| CCOBIX[2:0] | FCCOB Parameters | |
|---|---|---|
| 000 | 0x20 | Not required |
| 001 | Number of 256 byte sectors for the D-Flash user partition (DFPART) | |
| 010 | Number of 256 byte sectors for buffer RAM EEE partition (ERPART) | |

Upon clearing CCIF to launch the Partition D-Flash command, the following actions are taken to define a partition within the D-Flash block for direct access (DFPART) and a partition within the buffer RAM for EEE use (ERPART):

- Validate the DFPART and ERPART values provided:
  - DFPART <= 128 (maximum number of 256 byte sectors in D-Flash block)
  - ERPART <= 16 (maximum number of 256 byte sectors in buffer RAM)
  - If ERPART > 0, 128 - DFPART >= 12 (minimum number of 256 byte sectors in the D-Flash block required to support EEE)
  - If ERPART > 0, ((128-DFPART)/ERPART) >= 8 (minimum ratio of D-Flash EEE space to buffer RAM EEE space to support EEE)
- Erase verify the D-Flash block and the EEE nonvolatile information register
- Program DFPART to the EEE nonvolatile information register at global address 0x12_0000 (see Table 25-7)

- Program a duplicate DFPART to the EEE nonvolatile information register at global address 0x12_0002 (see Table 25-7)
- Program ERPART to the EEE nonvolatile information register at global address 0x12_0004 (see Table 25-7)
- Program a duplicate ERPART to the EEE nonvolatile information register at global address 0x12_0006 (see Table 25-7)

The D-Flash user partition will start at global address 0x10_0000. The buffer RAM EEE partition will end at global address 0x13_FFFF. After the Partition D-Flash operation has completed, the CCIF flag will set.

Running the Partition D-Flash command a second time will result in the ACCERR bit within the FSTAT register being set. The data value written corresponds to the number of 256 byte sectors allocated for either direct D-Flash access (DFPART) or buffer RAM EEE access (ERPART).

**Table 25-78. Partition D-Flash Command Error Handling**

| Register | Error Bit | Error Condition |
|---|---|---|
| FSTAT | ACCERR | Set if CCOBIX[2:0] != 010 at command launch |
| | | Set if a Load Data Field command sequence is currently active |
| | | Set if command not available in current mode (see Table 25-30) |
| | | Set if partitions have already been defined |
| | | Set if an invalid DFPART or ERPART selection is supplied |
| | FPVIOL | None |
| | MGSTAT1 | Set if any errors have been encountered during the read |
| | MGSTAT0 | Set if any non-correctable errors have been encountered during the read |
| FERSTAT | EPVIOLIF | None |

## 25.4.3 Interrupts

The Flash module can generate an interrupt when a Flash command operation has completed or when a Flash command operation has detected an EEE error or an ECC fault.

**Table 25-79. Flash Interrupt Sources**

| Interrupt Source | Interrupt Flag | Local Enable | Global (CCR) Mask |
|---|---|---|---|
| Flash Command Complete | CCIF (FSTAT register) | CCIE (FCNFG register) | I Bit |
| Flash EEE Erase Error | ERSERIF (FERSTAT register) | ERSERIE (FERCNFG register) | I Bit |
| Flash EEE Program Error | PGMERIF (FERSTAT register) | PGMERIE (FERCNFG register) | I Bit |
| Flash EEE Protection Violation | EPVIOLIF (FERSTAT register) | EPVIOLIE (FERCNFG register) | I Bit |
| Flash EEE Error Type 1 Violation | ERSVIF1 (FERSTAT register) | ERSVIE1 (FERCNFG register) | I Bit |
| Flash EEE Error Type 0 Violation | ERSVIF0 (FERSTAT register) | ERSVIE0 (FERCNFG register) | I Bit |
| ECC Double Bit Fault on Flash Read | DFDIF (FERSTAT register) | DFDIE (FERCNFG register) | I Bit |
| ECC Single Bit Fault on Flash Read | SFDIF (FERSTAT register) | SFDIE (FERCNFG register) | I Bit |

**NOTE**

Vector addresses and their relative interrupt priority are determined at the MCU level.

### 25.4.3.1 Description of Flash Interrupt Operation

The Flash module uses the CCIF flag in combination with the CCIE interrupt enable bit to generate the Flash command interrupt request. The Flash module uses the ERSEIF, PGMEIF, EPVIOLIF, ERSVIF1, ERSVIF0, DFDIF and SFDIF flags in combination with the ERSEIE, PGMEIE, EPVIOLIE, ERSVIE1, ERSVIE0, DFDIE and SFDIE interrupt enable bits to generate the Flash error interrupt request. For a detailed description of the register bits involved, refer to Section 25.3.2.5, “Flash Configuration Register (FCNFG)”, Section 25.3.2.6, “Flash Error Configuration Register (FERCNFG)”, Section 25.3.2.7, “Flash Status Register (FSTAT)”, and Section 25.3.2.8, “Flash Error Status Register (FERSTAT)”.

The logic used for generating the Flash module interrupts is shown in Figure 25-27.

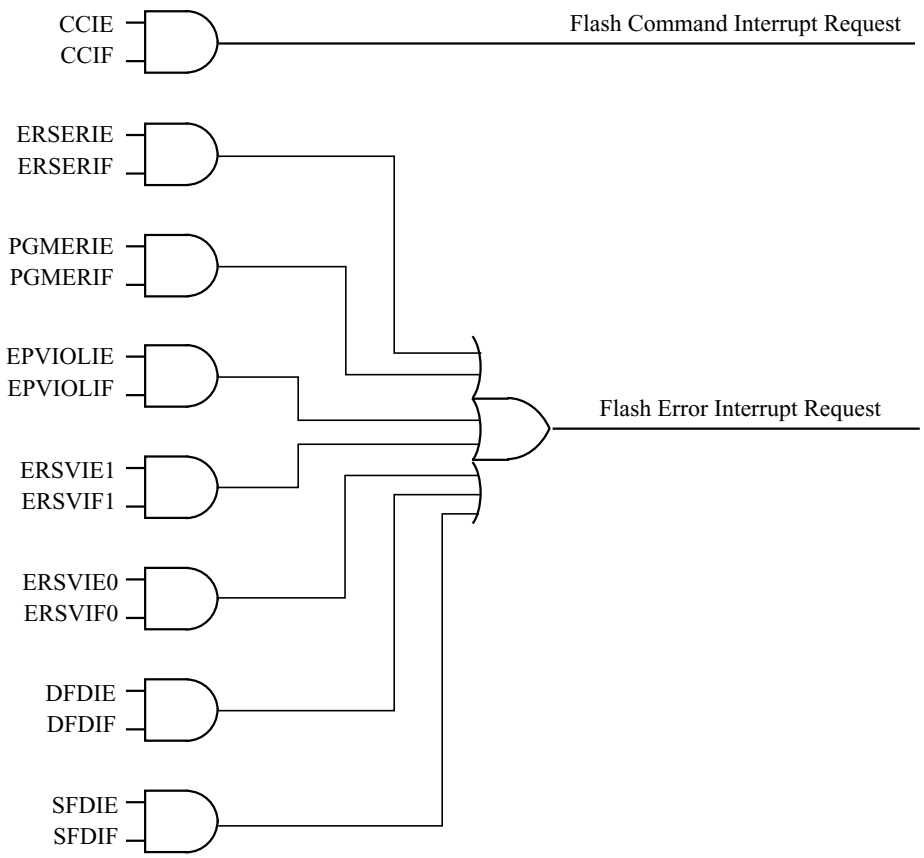



**Figure 25-27. Flash Module Interrupts Implementation**

### 25.4.4 Wait Mode

The Flash module is not affected if the MCU enters wait mode. The Flash module can recover the MCU from wait via the CCIF interrupt (see Section 25.4.3, “Interrupts”).

### 25.4.5 Stop Mode

If a Flash command is active (CCIF = 0) or an EE-Emulation operation is pending when the MCU requests stop mode, the current Flash operation will be completed before the CPU is allowed to enter stop mode.

## 25.5 Security

The Flash module provides security information to the MCU. The Flash security state is defined by the SEC bits of the FSEC register (see Table 25-12). During reset, the Flash module initializes the FSEC register using data read from the security byte of the Flash configuration field at global address 0x7F_FF0F.

The security state out of reset can be permanently changed by programming the security byte of the Flash configuration field. This assumes that you are starting from a mode where the necessary P-Flash erase and program commands are available and that the upper region of the P-Flash is unprotected. If the Flash security byte is successfully programmed, its new value will take affect after the next MCU reset.

The following subsections describe these security-related subjects:

- Unsecuring the MCU using Backdoor Key Access
- Unsecuring the MCU in Special Single Chip Mode using BDM
- Mode and Security Effects on Flash Command Availability

### 25.5.1 Unsecuring the MCU using Backdoor Key Access

The MCU may be unsecured by using the backdoor key access feature which requires knowledge of the contents of the backdoor keys (four 16-bit words programmed at addresses 0x7F_FF00–0x7F_FF07). If the KEYEN[1:0] bits are in the enabled state (see Section 25.3.2.2), the Verify Backdoor Access Key command (see Section 25.4.2.12) allows the user to present four prospective keys for comparison to the keys stored in the Flash memory via the Memory Controller. If the keys presented in the Verify Backdoor Access Key command match the backdoor keys stored in the Flash memory, the SEC bits in the FSEC register (see Table 25-12) will be changed to unsecure the MCU. Key values of 0x0000 and 0xFFFF are not permitted as backdoor keys. While the Verify Backdoor Access Key command is active, P-Flash block 0 will not be available for read access and will return invalid data.

The user code stored in the P-Flash memory must have a method of receiving the backdoor keys from an external stimulus. This external stimulus would typically be through one of the on-chip serial ports.

If the KEYEN[1:0] bits are in the enabled state (see Section 25.3.2.2), the MCU can be unsecured by the backdoor key access sequence described below:

1. Follow the command sequence for the Verify Backdoor Access Key command as explained in Section 25.4.2.12
2. If the Verify Backdoor Access Key command is successful, the MCU is unsecured and the SEC[1:0] bits in the FSEC register are forced to the unsecure state of 10

The Verify Backdoor Access Key command is monitored by the Memory Controller and an illegal key will prohibit future use of the Verify Backdoor Access Key command. A reset of the MCU is the only method to re-enable the Verify Backdoor Access Key command.

After the backdoor keys have been correctly matched, the MCU will be unsecured. After the MCU is unsecured, the sector containing the Flash security byte can be erased and the Flash security byte can be reprogrammed to the unsecure state, if desired.

In the unsecure state, the user has full control of the contents of the backdoor keys by programming addresses 0x7F_FF00–0x7F_FF07 in the Flash configuration field.

The security as defined in the Flash security byte (0x7F_FF0F) is not changed by using the Verify Backdoor Access Key command sequence. The backdoor keys stored in addresses 0x7F_FF00–0x7F_FF07 are unaffected by the Verify Backdoor Access Key command sequence. After the next reset of the MCU, the security state of the Flash module is determined by the Flash security byte

(0x7F_FF0F). The Verify Backdoor Access Key command sequence has no effect on the program and erase protections defined in the Flash protection register, FPROT.

### 25.5.2 Unsecuring the MCU in Special Single Chip Mode using BDM

The MCU can be unsecured in special single chip mode by erasing the P-Flash and D-Flash memory by one of the following methods:

- Reset the MCU into special single chip mode, delay while the erase test is performed by the BDM, send BDM commands to disable protection in the P-Flash and D-Flash memory, and execute the Erase All Blocks command write sequence to erase the P-Flash and D-Flash memory.
- Reset the MCU into special expanded wide mode, disable protection in the P-Flash and D-Flash memory and run code from external memory to execute the Erase All Blocks command write sequence to erase the P-Flash and D-Flash memory.

After the CCIF flag sets to indicate that the Erase All Blocks operation has completed, reset the MCU into special single chip mode. The BDM will execute the Erase Verify All Blocks command write sequence to verify that the P-Flash and D-Flash memory is erased. If the P-Flash and D-Flash memory are verified as erased the MCU will be unsecured. All BDM commands will be enabled and the Flash security byte may be programmed to the unsecure state by the following method:

- Send BDM commands to execute a ‘Program P-Flash’ command sequence to program the Flash security byte to the unsecured state and reset the MCU.

### 25.5.3 Mode and Security Effects on Flash Command Availability

The availability of Flash module commands depends on the MCU operating mode and security state as shown in Table 25-30.

## 25.6 Initialization

On each system reset the Flash module executes a reset sequence which establishes initial values for the Flash Block Configuration Parameters, the FPROT and DFPROT protection registers, and the FOPT and FSEC registers. The Flash module reverts to built-in default values that leave the module in a fully protected and secured state if errors are encountered during execution of the reset sequence. If a double bit fault is detected during the reset sequence, both MGSTAT bits in the FSTAT register will be set. The ACCERR bit in the FSTAT register is set if errors are encountered while initializing the EEE buffer ram during the reset sequence.

CCIF remains clear throughout the reset sequence. The Flash module holds off all CPU access for the initial portion of the reset sequence. While Flash reads are possible when the hold is removed, writes to the FCCOBIX, FCCOBHI, and FCCOBLO registers are ignored to prevent command activity while the Memory Controller remains busy. Completion of the reset sequence is marked by setting CCIF high which enables writes to the FCCOBIX, FCCOBHI, and FCCOBLO registers to launch any available Flash command.

If a reset occurs while any Flash command is in progress, that command will be immediately aborted. The state of the word being programmed or the sector/block being erased is not guaranteed.

# Chapter 26
# 384 KByte Flash Module (S12XFTM384K2V1)

**Table 26-1. Revision History**

| Revision Number | Revision Date | Sections Affected | Description of Changes |
|---|---|---|---|
| V01.10 | 29 Nov 2007 | | - Cleanup |
| V01.11 | 19 Dec 2007 | 26.4.2/26-989<br><br>26.4.2/26-989<br><br>26.3.1/26-958 | - Updated Command Error Handling tables based on parent-child relationship with FTM512K3<br>- Corrected Error Handling table for Full Partition D-Flash, Partition D-Flash, and EEPROM Emulation Query commands<br>- Corrected P-Flash IFR Accessibility table |
| V01.12 | 25 Sep 2009 | 26.1/26-953<br>26.3.2.1/26-965<br>26.4.2.4/26-992<br><br>26.4.2.7/26-995<br><br>26.4.2.12/26-999<br><br><br>26.4.2.12/26-999<br>26.4.2.12/26-999<br>26.4.2.20/26-1008<br><br><br><br>26.3.2/26-963<br>26.3.2.1/26-965<br>26.4.1.2/26-984<br>26.6/26-1014 | - Clarify single bit fault correction for P-Flash phrase<br>- Expand FDIV vs OSCCLK Frequency table<br>- Add statement concerning code runaway when executing Read Once command from Flash block containing associated fields<br>- Add statement concerning code runaway when executing Program Once command from Flash block containing associated fields<br>- Add statement concerning code runaway when executing Verify Backdoor Access Key command from Flash block containing associated fields<br>- Relate Key 0 to associated Backdoor Comparison Key address<br>- Change “power down reset” to “reset”<br>- Add ACCERR condition for Disable EEPROM Emulation command<br>The following changes were made to clarify module behavior related to Flash register access during reset sequence and while Flash commands are active:<br>- Add caution concerning register writes while command is active<br>- Writes to FCLKDIV are allowed during reset sequence while CCIF is clear<br>- Add caution concerning register writes while command is active<br>- Writes to FCCOBIX, FCCOBHI, FCCOBLO registers are ignored during reset sequence |

## 26.1 Introduction

The FTM384K2 module implements the following:

- 384 Kbytes of P-Flash (Program Flash) memory, consisting of 2 physical Flash blocks, intended primarily for nonvolatile code storage
- 32 Kbytes of D-Flash (Data Flash) memory, consisting of 1 physical Flash block, that can be used as nonvolatile storage to support the built-in hardware scheme for emulated EEPROM, as basic Flash memory primarily intended for nonvolatile data storage, or as a combination of both

- 4 Kbytes of buffer RAM, consisting of 1 physical RAM block, that can be used as emulated EEPROM using a built-in hardware scheme, as basic RAM, or as a combination of both

The Flash memory is ideal for single-supply applications allowing for field reprogramming without requiring external high voltage sources for program or erase operations. The Flash module includes a memory controller that executes commands to modify Flash memory contents or configure module resources for emulated EEPROM operation. The user interface to the memory controller consists of the indexed Flash Common Command Object (FCCOB) register which is written to with the command, global address, data, and any required command parameters. The memory controller must complete the execution of a command before the FCCOB register can be written to with a new command.

**CAUTION**

A Flash word or phrase must be in the erased state before being programmed. Cumulative programming of bits within a Flash word or phrase is not allowed.

The RAM and Flash memory may be read as bytes, aligned words, or misaligned words. Read access time is one bus cycle for bytes and aligned words, and two bus cycles for misaligned words. For Flash memory, an erased bit reads 1 and a programmed bit reads 0.

It is not possible to read from a Flash block while any command is executing on that specific Flash block. It is possible to read from a Flash block while a command is executing on a different Flash block.

Both P-Flash and D-Flash memories are implemented with Error Correction Codes (ECC) that can resolve single bit faults and detect double bit faults. For P-Flash memory, the ECC implementation requires that programming be done on an aligned 8 byte basis (a Flash phrase). Since P-Flash memory is always read by phrase, only one single bit fault in the phrase containing the byte or word accessed will be corrected.

### 26.1.1 Glossary

**Buffer RAM** — The buffer RAM constitutes the volatile memory store required for EEE. Memory space in the buffer RAM not required for EEE can be partitioned to provide volatile memory space for applications.

**Command Write Sequence** — An MCU instruction sequence to execute built-in algorithms (including program and erase) on the Flash memory.

**D-Flash Memory** — The D-Flash memory constitutes the nonvolatile memory store required for EEE. Memory space in the D-Flash memory not required for EEE can be partitioned to provide nonvolatile memory space for applications.

**D-Flash Sector** — The D-Flash sector is the smallest portion of the D-Flash memory that can be erased. The D-Flash sector consists of four 64 byte rows for a total of 256 bytes.

**EEE (Emulated EEPROM)** — A method to emulate the small sector size features and endurance characteristics associated with an EEPROM.

**EEE IFR** — Nonvolatile information register located in the D-Flash block that contains data required to partition the D-Flash memory and buffer RAM for EEE. The EEE IFR is visible in the global memory map by setting the EEEIFRON bit in the MMCCTL1 register.

**NVM Command Mode** — An NVM mode using the CPU to setup the FCCOB register to pass parameters required for Flash command execution.

**Phrase** — An aligned group of four 16-bit words within the P-Flash memory. Each phrase includes eight ECC bits for single bit fault correction and double bit fault detection within the phrase.

**P-Flash Memory** — The P-Flash memory constitutes the main nonvolatile memory store for applications.

**P-Flash Sector** — The P-Flash sector is the smallest portion of the P-Flash memory that can be erased. Each P-Flash sector contains 1024 bytes.

**Program IFR** — Nonvolatile information register located in the P-Flash block that contains the Device ID, Version ID, and the Program Once field. The Program IFR is visible in the global memory map by setting the PGMIFRON bit in the MMCCTL1 register.

## 26.1.2 Features

### 26.1.2.1 P-Flash Features

- 384 Kbytes of P-Flash memory composed of one 256 Kbyte Flash block and one 128 Kbyte Flash block. The 256 Kbyte Flash block consists of two 128 Kbyte sections each divided into 128 sectors of 1024 bytes. The 128 Kbyte Flash block is divided into 128 sectors of 1024 bytes.
- Single bit fault correction and double bit fault detection within a 64-bit phrase during read operations
- Automated program and erase algorithm with verify and generation of ECC parity bits
- Fast sector erase and phrase program operation
- Ability to program up to one phrase in each P-Flash block simultaneously
- Flexible protection scheme to prevent accidental program or erase of P-Flash memory

### 26.1.2.2 D-Flash Features

- Up to 32 Kbytes of D-Flash memory with 256 byte sectors for user access
- Dedicated commands to control access to the D-Flash memory over EEE operation
- Single bit fault correction and double bit fault detection within a word during read operations
- Automated program and erase algorithm with verify and generation of ECC parity bits
- Fast sector erase and word program operation
- Ability to program up to four words in a burst sequence

#### 26.1.2.3 Emulated EEPROM Features

- Up to 4 Kbytes of emulated EEPROM (EEE) accessible as 4 Kbytes of RAM
- Flexible protection scheme to prevent accidental program or erase of data
- Automatic EEE file handling using an internal Memory Controller
- Automatic transfer of valid EEE data from D-Flash memory to buffer RAM on reset
- Ability to monitor the number of outstanding EEE related buffer RAM words left to be programmed into D-Flash memory
- Ability to disable EEE operation and allow priority access to the D-Flash memory
- Ability to cancel all pending EEE operations and allow priority access to the D-Flash memory

#### 26.1.2.4 User Buffer RAM Features

- Up to 4 Kbytes of RAM for user access

#### 26.1.2.5 Other Flash Module Features

- No external high-voltage power supply required for Flash memory program and erase operations
- Interrupt generation on Flash command completion and Flash error detection
- Security mechanism to prevent unauthorized access to the Flash memory

### 26.1.3 Block Diagram

The block diagram of the Flash module is shown in Figure 26-1.

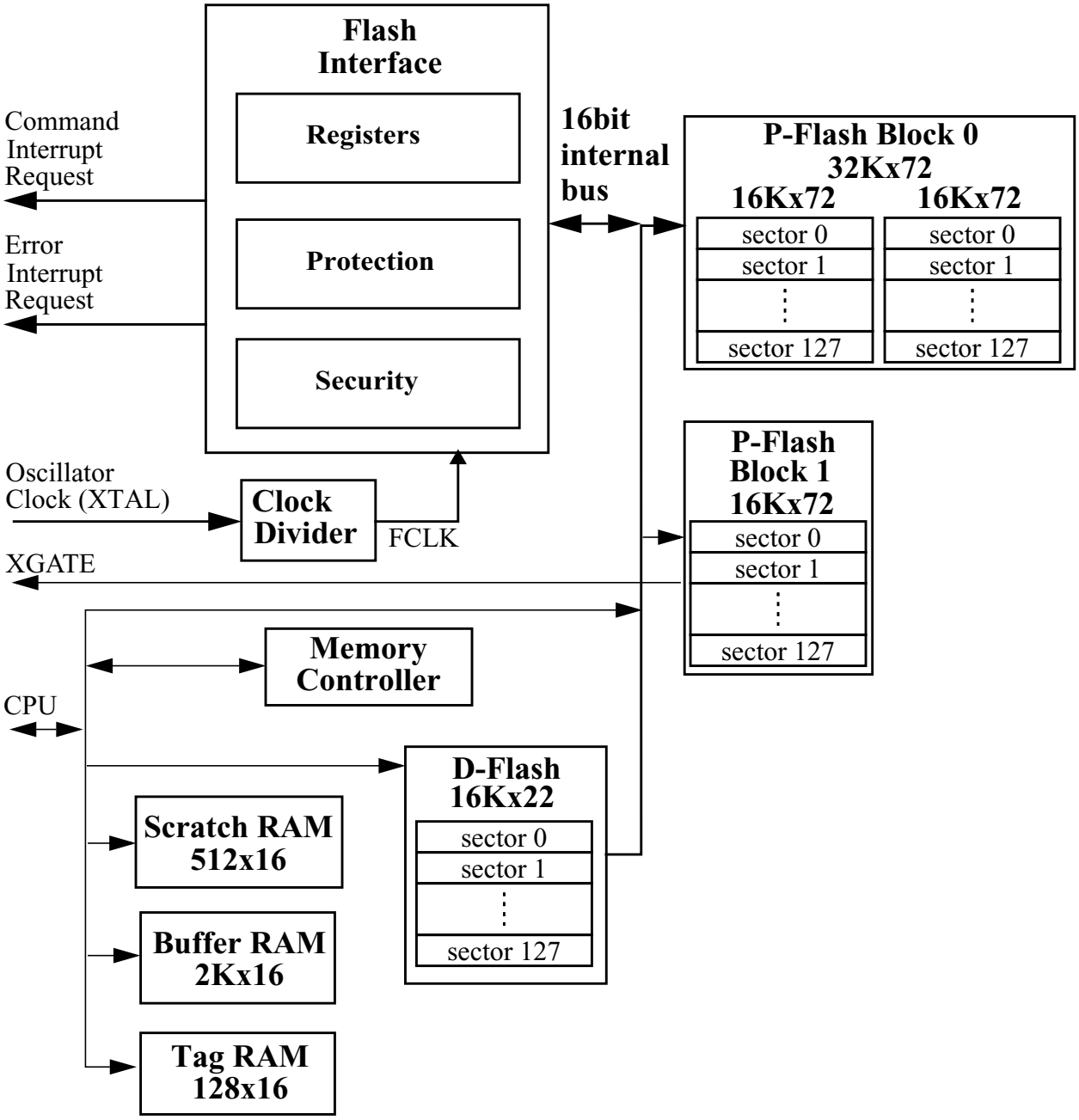



**Figure 26-1. FTM384K2 Block Diagram**

## 26.2 External Signal Description

The Flash module contains no signals that connect off-chip.

## 26.3 Memory Map and Registers

This section describes the memory map and registers for the Flash module. Read data from unimplemented memory space in the Flash module is undefined. Write access to unimplemented or reserved memory space in the Flash module will be ignored by the Flash module.

### 26.3.1 Module Memory Map

The S12X architecture places the P-Flash memory between global addresses 0x78_0000 and 0x7F_FFFF as shown in Table 26-2. The P-Flash memory map is shown in Figure 26-2.

**Table 26-2. P-Flash Memory Addressing**

| Global Address | Size (Bytes) | Description |
|---|---|---|
| 0x7C_0000 – 0x7F_FFFF | 256 K | P-Flash Block 0<br>Contains Flash Configuration Field (see Table 26-3) |
| 0x7A_0000 – 0x7B_FFFF | 128 K | No P-Flash Memory |
| 0x78_0000 – 0x79_FFFF | 128 K | P-Flash Block 1 |

The FPROT register, described in Section 26.3.2.9, can be set to protect regions in the Flash memory from accidental program or erase. Three separate memory regions, one growing upward from global address 0x7F_8000 in the Flash memory (called the lower region), one growing downward from global address 0x7F_FFFF in the Flash memory (called the higher region), and the remaining addresses in the Flash memory, can be activated for protection. The Flash memory addresses covered by these protectable regions are shown in the P-Flash memory map. The higher address region is mainly targeted to hold the boot loader code since it covers the vector space. Default protection settings as well as security information that allows the MCU to restrict access to the Flash module are stored in the Flash configuration field as described in Table 26-3.

**Table 26-3. Flash Configuration Field(1)**

| Global Address | Size (Bytes) | Description |
|---|---|---|
| 0x7F_FF00 – 0x7F_FF07 | 8 | Backdoor Comparison Key<br>Refer to Section 26.4.2.12, “Verify Backdoor Access Key Command,” and Section 26.5.1, “Unsecuring the MCU using Backdoor Key Access” |
| 0x7F_FF08 – 0x7F_FF0B(2) | 4 | Reserved |
| 0x7F_FF0C[2] | 1 | P-Flash Protection byte.<br>Refer to Section 26.3.2.9, “P-Flash Protection Register (FPROT)” |
| 0x7F_FF0D[2] | 1 | EEE Protection byte<br>Refer to Section 26.3.2.10, “EEE Protection Register (EPROT)” |
| 0x7F_FF0E[2] | 1 | Flash Nonvolatile byte<br>Refer to Section 26.3.2.14, “Flash Option Register (FOPT)” |

**Table 26-3. Flash Configuration Field[1]**

| Global Address | Size (Bytes) | Description |
|---|---|---|
| 0x7F_FF0F[2] | 1 | Flash Security byte<br>Refer to Section 26.3.2.2, “Flash Security Register (FSEC)” |

1. Older versions may have swapped protection byte addresses
2. 0x7FF08 - 0x7F_FF0F form a Flash phrase and must be programmed in a single command write sequence. Each byte in the 0x7F_FF08 - 0x7F_FF0B reserved field should be programmed to 0xFF.

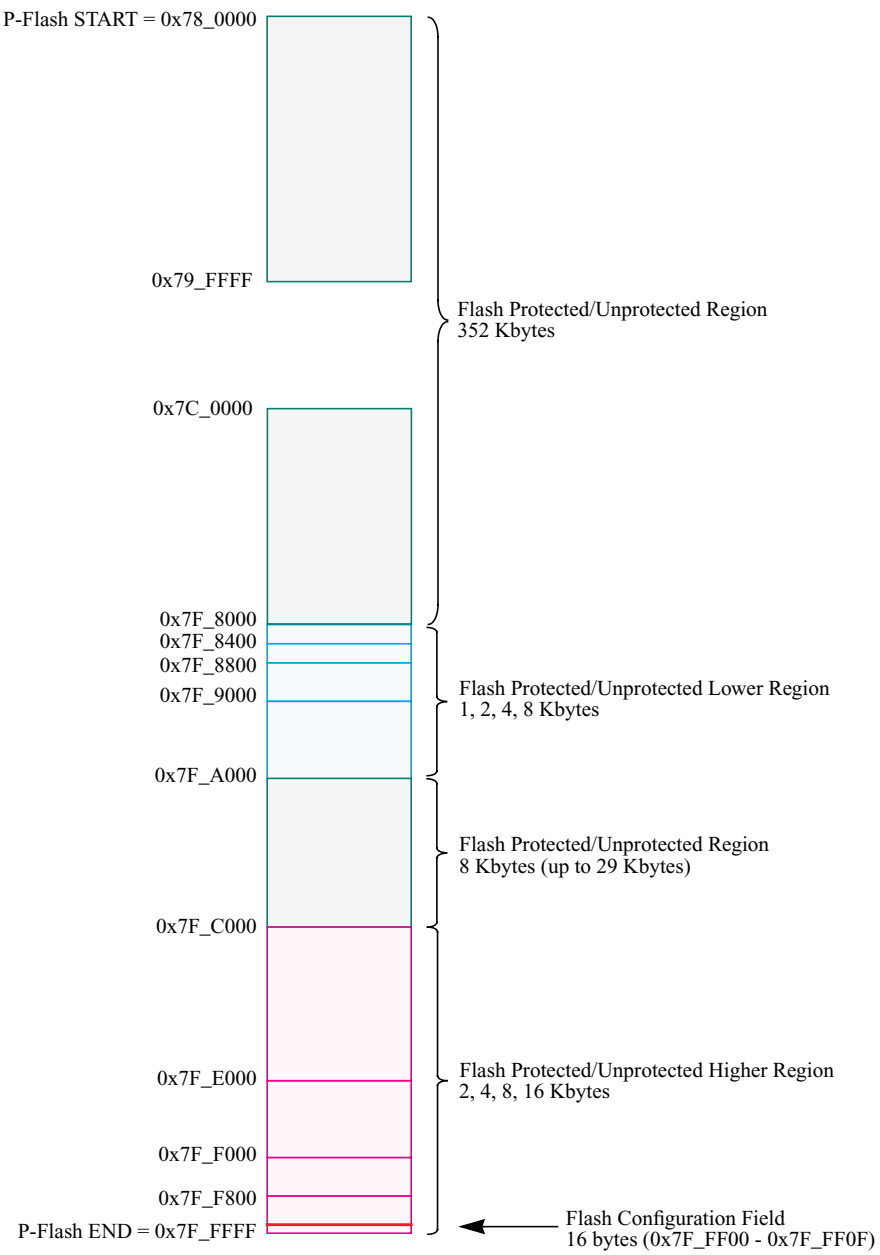



**Figure 26-2. P-Flash Memory Map**

**Table 26-4. Program IFR Fields**

| Global Address (PGMIFRON) | Size (Bytes) | Field Description |
|---|---|---|
| 0x40_0000 – 0x40_0007 | 8 | Device ID |
| 0x40_0008 – 0x40_00E7 | 224 | Reserved |
| 0x40_00E8 – 0x40_00E9 | 2 | Version ID |
| 0x40_00EA – 0x40_00FF | 22 | Reserved |
| 0x40_0100 – 0x40_013F | 64 | Program Once Field<br>Refer to Section 26.4.2.7, “Program Once Command” |
| 0x40_0140 – 0x40_01FF | 192 | Reserved |

**Table 26-5. P-Flash IFR Accessibility**

| Global Address (PGMIFRON) | Size (Bytes) | Accessed From |
|---|---|---|
| 0x40_0000 – 0x40_01FF | 512 | XBUS0 (PBLK0)(1) |
| 0x40_0200 – 0x40_03FF | 512 | Unimplemented |
| 0x40_0400 – 0x40_05FF | 512 | Unimplemented |
| 0x40_0600 – 0x40_07FF | 512 | XBUS1 (PBLK1) |

1. Refer to Table 26-4 for more details.

**Table 26-6. EEE Resource Fields**

| Global Address | Size (Bytes) | Description |
|---|---|---|
| 0x10_0000 – 0x10_7FFF | 32,768 | D-Flash Memory (User and EEE) |
| 0x10_8000 – 0x11_FFFF | 98,304 | Reserved |
| 0x12_0000 – 0x12_007F | 128 | EEE Nonvolatile Information Register (EEEIFRON(1) = 1) |
| 0x12_0080 – 0x12_0FFF | 3,968 | Reserved |
| 0x12_1000 – 0x12_1EFF | 3,840 | Reserved |
| 0x12_1F00 – 0x12_1FFF | 256 | EEE Tag RAM (TMGRAMON[1] = 1) |
| 0x12_2000 – 0x12_3BFF | 7,168 | Reserved |
| 0x12_3C00 – 0x12_3FFF | 1,024 | Memory Controller Scratch RAM (TMGRAMON[1] = 1) |
| 0x12_4000 – 0x12_DFFF | 40,960 | Reserved |
| 0x12_E000 – 0x12_FFFF | 8,192 | Reserved |
| 0x13_0000 – 0x13_EFFF | 61,440 | Reserved |
| 0x13_F000 – 0x13_FFFF | 4,096 | Buffer RAM (User and EEE) |

1. MMCCTL1 register bit

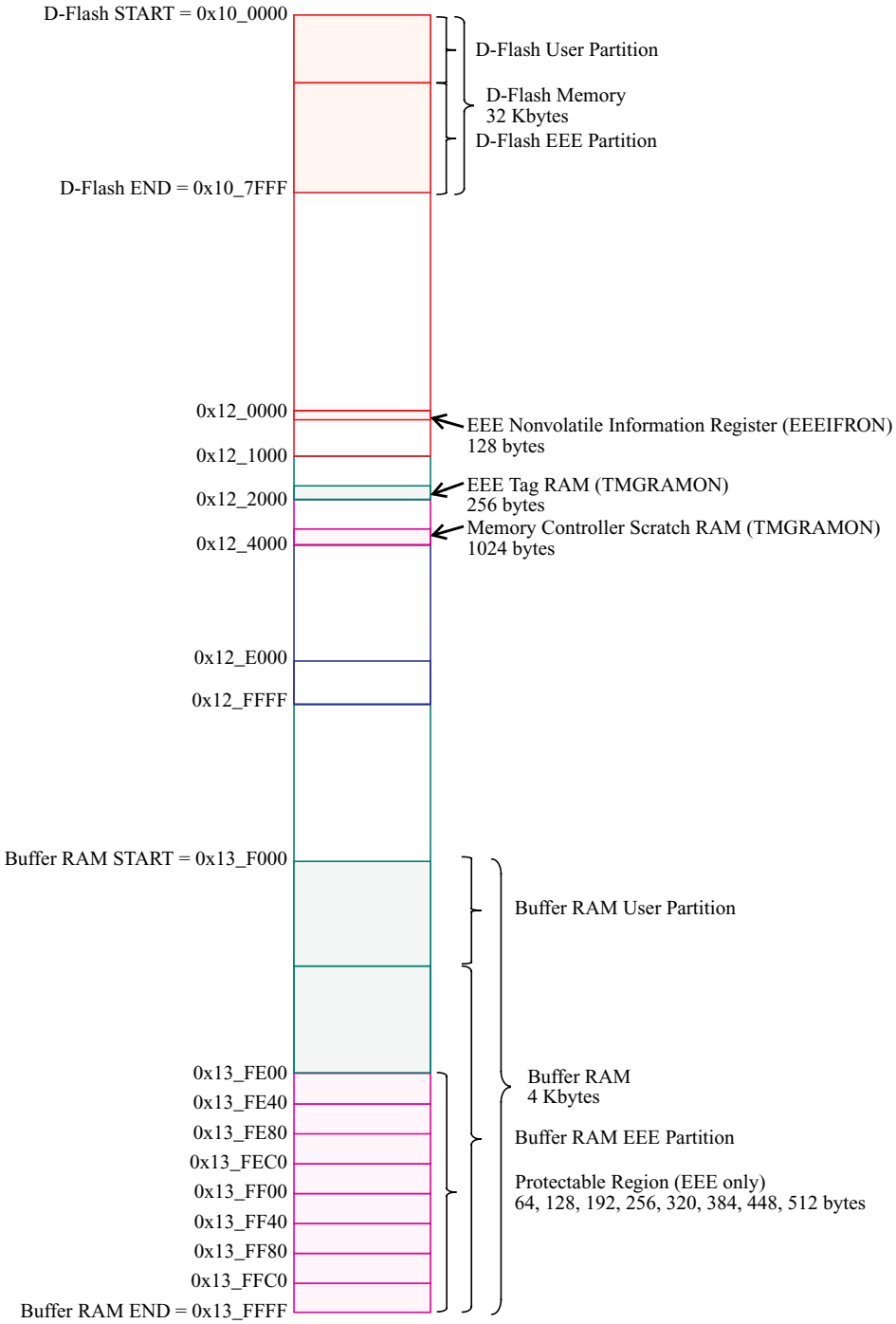



**Figure 26-3. EEE Resource Memory Map**

The Full Partition D-Flash command (see Section 26.4.2.15) is used to program the EEE nonvolatile information register fields where address 0x12_0000 defines the D-Flash partition for user access and address 0x12_0004 defines the buffer RAM partition for EEE operations.

**Table 26-7. EEE Nonvolatile Information Register Fields**

| Global Address (EEEIFRON) | Size (Bytes) | Description |
|---|---|---|
| 0x12_0000 – 0x12_0001 | 2 | D-Flash User Partition (DFPART)<br>Refer to Section 26.4.2.15, “Full Partition D-Flash Command” |
| 0x12_0002 – 0x12_0003 | 2 | D-Flash User Partition (duplicate(1)) |
| 0x12_0004 – 0x12_0005 | 2 | Buffer RAM EEE Partition (ERPART)<br>Refer to Section 26.4.2.15, “Full Partition D-Flash Command” |
| 0x12_0006 – 0x12_0007 | 2 | Buffer RAM EEE Partition (duplicate1) |
| 0x12_0008 – 0x12_007F | 120 | Reserved |

1. Duplicate value used if primary value generates a double bit fault when read during the reset sequence.

## 26.3.2 Register Descriptions

The Flash module contains a set of 20 control and status registers located between Flash module base + 0x0000 and 0x0013. A summary of the Flash module registers is given in Figure 26-4 with detailed descriptions in the following subsections.

**CAUTION**

Writes to any Flash register must be avoided while a Flash command is active (CCIF=0) to prevent corruption of Flash register contents and Memory Controller behavior.

| Address & Name | | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|
| 0x0000 FCLKDIV | R | FDIVLD | FDIV6 | FDIV5 | FDIV4 | FDIV3 | FDIV2 | FDIV1 | FDIV0 |
| | W | | | | | | | | |
| 0x0001 FSEC | R | KEYEN1 | KEYEN0 | RNV5 | RNV4 | RNV3 | RNV2 | SEC1 | SEC0 |
| | W | | | | | | | | |
| 0x0002 FCCOBIX | R | 0 | 0 | 0 | 0 | 0 | CCOBIX2 | CCOBIX1 | CCOBIX0 |
| | W | | | | | | | | |
| 0x0003 FECCRIX | R | 0 | 0 | 0 | 0 | 0 | ECCRIX2 | ECCRIX1 | ECCRIX0 |
| | W | | | | | | | | |
| 0x0004 FCNFG | R | CCIE | 0 | 0 | IGNSF | 0 | 0 | FDFD | FSFD |
| | W | | | | | | | | |
| 0x0005 FERCNFG | R | ERSERIE | PGMERIE | 0 | EPVIOLIE | ERSVIE1 | ERSVIE0 | DFDIE | SFDIE |
| | W | | | | | | | | |

**Figure 26-4. FTM384K2 Register Summary**

| Address & Name | | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|
| 0x0006 FSTAT | R | CCIF | 0 | ACCERR | FPVIOL | MGBUSY | RSVD | MGSTAT1 | MGSTAT0 |
| | W | | | | | | | | |
| 0x0007 FERSTAT | R | ERSERIF | PGMERIF | 0 | EPVIOLIF | ERSVIF1 | ERSVIF0 | DFDIF | SFDIF |
| | W | | | | | | | | |
| 0x0008 FPROT | R | FPOPEN | RNV6 | FPHDIS | FPHS1 | FPHS0 | FPLDIS | FPLS1 | FPLS0 |
| | W | | | | | | | | |
| 0x0009 EPROT | R | EPOPEN | RNV6 | RNV5 | RNV4 | EPDIS | EPS2 | EPS1 | EPS0 |
| | W | | | | | | | | |
| 0x000A FCCOBHI | R | CCOB15 | CCOB14 | CCOB13 | CCOB12 | CCOB11 | CCOB10 | CCOB9 | CCOB8 |
| | W | | | | | | | | |
| 0x000B FCCOBLO | R | CCOB7 | CCOB6 | CCOB5 | CCOB4 | CCOB3 | CCOB2 | CCOB1 | CCOB0 |
| | W | | | | | | | | |
| 0x000C ETAGHI | R | ETAG15 | ETAG14 | ETAG13 | ETAG12 | ETAG11 | ETAG10 | ETAG9 | ETAG8 |
| | W | | | | | | | | |
| 0x000D ETAGLO | R | ETAG7 | ETAG6 | ETAG5 | ETAG4 | ETAG3 | ETAG2 | ETAG1 | ETAG0 |
| | W | | | | | | | | |
| 0x000E FECCRHI | R | ECCR15 | ECCR14 | ECCR13 | ECCR12 | ECCR11 | ECCR10 | ECCR9 | ECCR8 |
| | W | | | | | | | | |
| 0x000F FECCRLO | R | ECCR7 | ECCR6 | ECCR5 | ECCR4 | ECCR3 | ECCR2 | ECCR1 | ECCR0 |
| | W | | | | | | | | |
| 0x0010 FOPT | R | NV7 | NV6 | NV5 | NV4 | NV3 | NV2 | NV1 | NV0 |
| | W | | | | | | | | |
| 0x0011 FRSV0 | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | W | | | | | | | | |
| 0x0012 FRSV1 | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | W | | | | | | | | |
| 0x0013 FRSV2 | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | W | | | | | | | | |

**Figure 26-4. FTM384K2 Register Summary (continued)**

**Figure 26-4. FTM384K2 Register Summary (continued)**

### 26.3.2.1 Flash Clock Divider Register (FCLKDIV)

The FCLKDIV register is used to control timed events in program and erase algorithms.

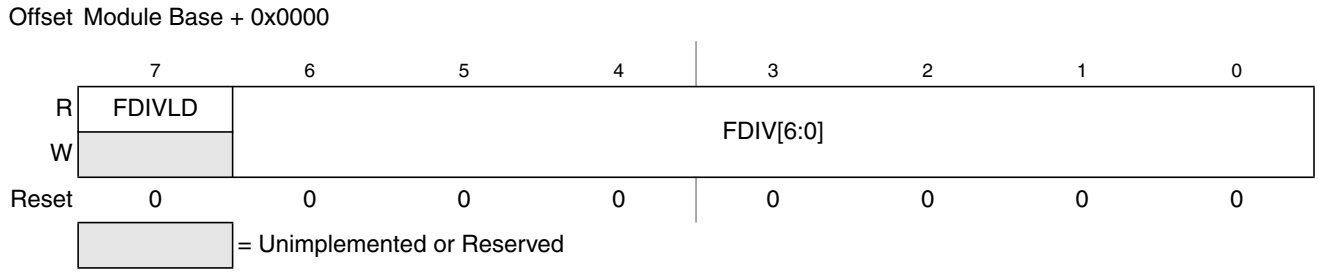



**Figure 26-5. Flash Clock Divider Register (FCLKDIV)**

All bits in the FCLKDIV register are readable, bits 6–0 are write once and bit 7 is not writable.

**Table 26-8. FCLKDIV Field Descriptions**

| Field | Description |
|---|---|
| 7<br>FDIVLD | **Clock Divider Loaded**<br>0 FCLKDIV register has not been written<br>1 FCLKDIV register has been written since the last reset |
| 6–0<br>FDIV[6:0] | **Clock Divider Bits** — FDIV[6:0] must be set to effectively divide OSCCLK down to generate an internal Flash clock, FCLK, with a target frequency of 1 MHz for use by the Flash module to control timed events during program and erase algorithms. Table 26-9 shows recommended values for FDIV[6:0] based on OSCCLK frequency. Please refer to Section 26.4.1, “Flash Command Operations,” for more information. |

**CAUTION**

The FCLKDIV register should never be written while a Flash command is executing (CCIF=0). The FCLKDIV register is writable during the Flash reset sequence even though CCIF is clear.

**Table 26-9. FDIV vs OSCCLK Frequency**

| OSCCLK Frequency (MHz) | | FDIV[6:0] | OSCCLK Frequency (MHz) | | FDIV[6:0] | OSCCLK Frequency (MHz) | | FDIV[6:0] |
|---|---|---|---|---|---|---|---|---|
| MIN(1) | MAX(2) | | MIN1 | MAX2 | | MIN1 | MAX2 | |
| | | | 33.60 | 34.65 | 0x20 | 67.20 | 68.25 | 0x40 |
| 1.60 | 2.10 | 0x01 | 34.65 | 35.70 | 0x21 | 68.25 | 69.30 | 0x41 |
| 2.40 | 3.15 | 0x02 | 35.70 | 36.75 | 0x22 | 69.30 | 70.35 | 0x42 |
| 3.20 | 4.20 | 0x03 | 36.75 | 37.80 | 0x23 | 70.35 | 71.40 | 0x43 |
| 4.20 | 5.25 | 0x04 | 37.80 | 38.85 | 0x24 | 71.40 | 72.45 | 0x44 |
| 5.25 | 6.30 | 0x05 | 38.85 | 39.90 | 0x25 | 72.45 | 73.50 | 0x45 |
| 6.30 | 7.35 | 0x06 | 39.90 | 40.95 | 0x26 | 73.50 | 74.55 | 0x46 |
| 7.35 | 8.40 | 0x07 | 40.95 | 42.00 | 0x27 | 74.55 | 75.60 | 0x47 |
| 8.40 | 9.45 | 0x08 | 42.00 | 43.05 | 0x28 | 75.60 | 76.65 | 0x48 |
| 9.45 | 10.50 | 0x09 | 43.05 | 44.10 | 0x29 | 76.65 | 77.70 | 0x49 |
| 10.50 | 11.55 | 0x0A | 44.10 | 45.15 | 0x2A | 77.70 | 78.75 | 0x4A |
| 11.55 | 12.60 | 0x0B | 45.15 | 46.20 | 0x2B | 78.75 | 79.80 | 0x4B |
| 12.60 | 13.65 | 0x0C | 46.20 | 47.25 | 0x2C | 79.80 | 80.85 | 0x4C |
| 13.65 | 14.70 | 0x0D | 47.25 | 48.30 | 0x2D | 80.85 | 81.90 | 0x4D |
| 14.70 | 15.75 | 0x0E | 48.30 | 49.35 | 0x2E | 81.90 | 82.95 | 0x4E |
| 15.75 | 16.80 | 0x0F | 49.35 | 50.40 | 0x2F | 82.95 | 84.00 | 0x4F |
| 16.80 | 17.85 | 0x10 | 50.40 | 51.45 | 0x30 | 84.00 | 85.05 | 0x50 |
| 17.85 | 18.90 | 0x11 | 51.45 | 52.50 | 0x31 | 85.05 | 86.10 | 0x51 |
| 18.90 | 19.95 | 0x12 | 52.50 | 53.55 | 0x32 | 86.10 | 87.15 | 0x52 |
| 19.95 | 21.00 | 0x13 | 53.55 | 54.60 | 0x33 | 87.15 | 88.20 | 0x53 |
| 21.00 | 22.05 | 0x14 | 54.60 | 55.65 | 0x34 | 88.20 | 89.25 | 0x54 |
| 22.05 | 23.10 | 0x15 | 55.65 | 56.70 | 0x35 | 89.25 | 90.30 | 0x55 |
| 23.10 | 24.15 | 0x16 | 56.70 | 57.75 | 0x36 | 90.30 | 91.35 | 0x56 |
| 24.15 | 25.20 | 0x17 | 57.75 | 58.80 | 0x37 | 91.35 | 92.40 | 0x57 |
| 25.20 | 26.25 | 0x18 | 58.80 | 59.85 | 0x38 | 92.40 | 93.45 | 0x58 |
| 26.25 | 27.30 | 0x19 | 59.85 | 60.90 | 0x39 | 93.45 | 94.50 | 0x59 |
| 27.30 | 28.35 | 0x1A | 60.90 | 61.95 | 0x3A | 94.50 | 95.55 | 0x5A |
| 28.35 | 29.40 | 0x1B | 61.95 | 63.00 | 0x3B | 95.55 | 96.60 | 0x5B |
| 29.40 | 30.45 | 0x1C | 63.00 | 64.05 | 0x3C | 96.60 | 97.65 | 0x5C |
| 30.45 | 31.50 | 0x1D | 64.05 | 65.10 | 0x3D | 97.65 | 98.70 | 0x5D |
| 31.50 | 32.55 | 0x1E | 65.10 | 66.15 | 0x3E | 98.70 | 99.75 | 0x5E |
| 32.55 | 33.60 | 0x1F | 66.15 | 67.20 | 0x3F | 99.75 | 100.80 | 0x5F |

1. FDIV shown generates an FCLK frequency of >0.8 MHz

2. FDIV shown generates an FCLK frequency of 1.05 MHz

### 26.3.2.2 Flash Security Register (FSEC)

The FSEC register holds all bits associated with the security of the MCU and Flash module.

Offset Module Base + 0x0001

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | KEYEN[1:0] | | RNV[5:2] | | | | SEC[1:0] | |
| W | | | | | | | | |
| Reset | F | F | F | F | F | F | F | F |

= Unimplemented or Reserved

**Figure 26-6. Flash Security Register (FSEC)**

All bits in the FSEC register are readable but not writable.

During the reset sequence, the FSEC register is loaded with the contents of the Flash security byte in the Flash configuration field at global address 0x7F_FF0F located in P-Flash memory (see Table 26-3) as indicated by reset condition F in Figure 26-6. If a double bit fault is detected while reading the P-Flash phrase containing the Flash security byte during the reset sequence, all bits in the FSEC register will be set to leave the Flash module in a secured state with backdoor key access disabled.

**Table 26-10. FSEC Field Descriptions**

| Field | Description |
|---|---|
| 7–6<br>KEYEN[1:0] | **Backdoor Key Security Enable Bits** — The KEYEN[1:0] bits define the enabling of backdoor key access to the Flash module as shown in Table 26-11. |
| 5–2<br>RNV[5:2} | **Reserved Nonvolatile Bits** — The RNV bits should remain in the erased state for future enhancements. |
| 1–0<br>SEC[1:0] | **Flash Security Bits** — The SEC[1:0] bits define the security state of the MCU as shown in Table 26-12. If the Flash module is unsecured using backdoor key access, the SEC bits are forced to 10. |

**Table 26-11. Flash KEYEN States**

| KEYEN[1:0] | Status of Backdoor Key Access |
|---|---|
| 00 | DISABLED |
| 01 | DISABLED(1) |
| 10 | ENABLED |
| 11 | DISABLED |

1. Preferred KEYEN state to disable backdoor key access.

**Table 26-12. Flash Security States**

| SEC[1:0] | Status of Security |
|---|---|
| 00 | SECURED |
| 01 | SECURED(1) |
| 10 | UNSECURED |
| 11 | SECURED |

1. Preferred SEC state to set MCU to secured state.

The security function in the Flash module is described in Section 26.5.

### 26.3.2.3 Flash CCOB Index Register (FCCOBIX)

The FCCOBIX register is used to index the FCCOB register for Flash memory operations.

Offset Module Base + 0x0002

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | 0 | 0 | 0 | CCOBIX[2:0] | | |
| W | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented or Reserved

**Figure 26-7. FCCOB Index Register (FCCOBIX)**

CCOBIX bits are readable and writable while remaining bits read 0 and are not writable.

**Table 26-13. FCCOBIX Field Descriptions**

| Field | Description |
|---|---|
| 2–0 CCOBIX[1:0] | **Common Command Register Index**— The CCOBIX bits are used to select which word of the FCCOB register array is being read or written to. See Section 26.3.2.11, “Flash Common Command Object Register (FCCOB),” for more details. |

### 26.3.2.4 Flash ECCR Index Register (FECCRIX)

The FECCRIX register is used to index the FECCR register for ECC fault reporting.

Offset Module Base + 0x0003

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | 0 | 0 | 0 | ECCRIX[2:0] | | |
| W | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented or Reserved

**Figure 26-8. FECCR Index Register (FECCRIX)**

ECCRIX bits are readable and writable while remaining bits read 0 and are not writable.

**Table 26-14. FECCRIX Field Descriptions**

| Field | Description |
|---|---|
| 2-0<br>ECCRIX[2:0] | **ECC Error Register Index**— The ECCRIX bits are used to select which word of the FECCR register array is being read. See Section 26.3.2.13, “Flash ECC Error Results Register (FECCR),” for more details. |

### 26.3.2.5 Flash Configuration Register (FCNFG)

The FCNFG register enables the Flash command complete interrupt and forces ECC faults on Flash array read access from the CPU or XGATE.

Offset Module Base + 0x0004

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | CCIE | 0 | 0 | IGNSF | 0 | 0 | FDFD | FSFD |
| W | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented or Reserved

**Figure 26-9. Flash Configuration Register (FCNFG)**

CCIE, IGNSF, FDFD, and FSFD bits are readable and writable while remaining bits read 0 and are not writable.

**Table 26-15. FCNFG Field Descriptions**

| Field | Description |
|---|---|
| 7<br>CCIE | **Command Complete Interrupt Enable** — The CCIE bit controls interrupt generation when a Flash command has completed.<br>0 Command complete interrupt disabled<br>1 An interrupt will be requested whenever the CCIF flag in the FSTAT register is set (see Section 26.3.2.7) |
| 4<br>IGNSF | **Ignore Single Bit Fault** — The IGNSF controls single bit fault reporting in the FERSTAT register (see Section 26.3.2.8).<br>0 All single bit faults detected during array reads are reported<br>1 Single bit faults detected during array reads are not reported and the single bit fault interrupt will not be generated |

**Table 26-15. FCNFG Field Descriptions (continued)**

| Field | Description |
|---|---|
| 1<br>FDFD | **Force Double Bit Fault Detect** — The FDFD bit allows the user to simulate a double bit fault during Flash array read operations and check the associated interrupt routine. The FDFD bit is cleared by writing a 0 to FDFD. The FECCR registers will not be updated during the Flash array read operation with FDFD set unless an actual double bit fault is detected.<br>0 Flash array read operations will set the DFDIF flag in the FERSTAT register only if a double bit fault is detected<br>1 Any Flash array read operation will force the DFDIF flag in the FERSTAT register to be set (see Section 26.3.2.7) and an interrupt will be generated as long as the DFDIE interrupt enable in the FERCNFG register is set (see Section 26.3.2.6) |
| 0<br>FSFD | **Force Single Bit Fault Detect** — The FSFD bit allows the user to simulate a single bit fault during Flash array read operations and check the associated interrupt routine. The FSFD bit is cleared by writing a 0 to FSFD. The FECCR registers will not be updated during the Flash array read operation with FSFD set unless an actual single bit fault is detected.<br>0 Flash array read operations will set the SFDIF flag in the FERSTAT register only if a single bit fault is detected<br>1 Flash array read operation will force the SFDIF flag in the FERSTAT register to be set (see Section 26.3.2.7) and an interrupt will be generated as long as the SFDIE interrupt enable in the FERCNFG register is set (see Section 26.3.2.6) |

## 26.3.2.6 Flash Error Configuration Register (FERCNFG)

The FERCNFG register enables the Flash error interrupts for the FERSTAT flags.

Offset Module Base + 0x0005

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | ERSERIE | PGMERIE | 0 | EPVIOLIE | ERSVIE1 | ERSVIE0 | DFDIE | SFDIE |
| W | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented or Reserved

**Figure 26-10. Flash Error Configuration Register (FERCNFG)**

All assigned bits in the FERCNFG register are readable and writable.

**Table 26-16. FERCNFG Field Descriptions**

| Field | Description |
|---|---|
| 7<br>ERSERIE | **EEE Erase Error Interrupt Enable** — The ERSERIE bit controls interrupt generation when a failure is detected during an EEE erase operation.<br>0 ERSERIF interrupt disabled<br>1 An interrupt will be requested whenever the ERSERIF flag is set (see Section 26.3.2.8) |
| 6<br>PGMERIE | **EEE Program Error Interrupt Enable** — The PGMERIE bit controls interrupt generation when a failure is detected during an EEE program operation.<br>0 PGMERIF interrupt disabled<br>1 An interrupt will be requested whenever the PGMERIF flag is set (see Section 26.3.2.8) |
| 4<br>EPVIOLIE | **EEE Protection Violation Interrupt Enable** — The EPVIOLIE bit controls interrupt generation when a protection violation is detected during a write to the buffer RAM EEE partition.<br>0 EPVIOLIF interrupt disabled<br>1 An interrupt will be requested whenever the EPVIOLIF flag is set (see Section 26.3.2.8) |

**Table 26-16. FERCNFG Field Descriptions (continued)**

| Field | Description |
|---|---|
| 3<br>ERSVIE1 | **EEE Error Type 1 Interrupt Enable** — The ERSVIE1 bit controls interrupt generation when a change state error is detected during an EEE operation.<br>0 ERSVIF1 interrupt disabled<br>1 An interrupt will be requested whenever the ERSVIF1 flag is set (see Section 26.3.2.8) |
| 2<br>ERSVIE0 | **EEE Error Type 0 Interrupt Enable** — The ERSVIE0 bit controls interrupt generation when a sector format error is detected during an EEE operation.<br>0 ERSVIF0 interrupt disabled<br>1 An interrupt will be requested whenever the ERSVIF0 flag is set (see Section 26.3.2.8) |
| 1<br>DFDIE | **Double Bit Fault Detect Interrupt Enable** — The DFDIE bit controls interrupt generation when a double bit fault is detected during a Flash block read operation.<br>0 DFDIF interrupt disabled<br>1 An interrupt will be requested whenever the DFDIF flag is set (see Section 26.3.2.8) |
| 0<br>SFDIE | **Single Bit Fault Detect Interrupt Enable** — The SFDIE bit controls interrupt generation when a single bit fault is detected during a Flash block read operation.<br>0 SFDIF interrupt disabled whenever the SFDIF flag is set (see Section 26.3.2.8)<br>1 An interrupt will be requested whenever the SFDIF flag is set (see Section 26.3.2.8) |

## 26.3.2.7 Flash Status Register (FSTAT)

The FSTAT register reports the operational status of the Flash module.

Offset Module Base + 0x0006

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | CCIF | 0 | ACCERR | FPVIOL | MGBUSY | RSVD | MGSTAT[1:0] | |
| W | | | | | | | | |
| Reset | 1 | 0 | 0 | 0 | 0 | 0 | 0(1) | 0[1] |

= Unimplemented or Reserved

**Figure 26-11. Flash Status Register (FSTAT)**

1. Reset value can deviate from the value shown if a double bit fault is detected during the reset sequence (see Section 26.6).

CCIF, ACCERR, and FPVIOL bits are readable and writable, MGBUSY and MGSTAT bits are readable but not writable, while remaining bits read 0 and are not writable.

**Table 26-17. FSTAT Field Descriptions**

| Field | Description |
|---|---|
| 7<br>CCIF | **Command Complete Interrupt Flag** — The CCIF flag indicates that a Flash command has completed. The CCIF flag is cleared by writing a 1 to CCIF to launch a command and CCIF will stay low until command completion or command violation.<br>0 Flash command in progress<br>1 Flash command has completed |
| 5<br>ACCERR | **Flash Access Error Flag** — The ACCERR bit indicates an illegal access has occurred to the Flash memory caused by either a violation of the command write sequence (see Section 26.4.1.2) or issuing an illegal Flash command or when errors are encountered while initializing the EEE buffer ram during the reset sequence. While ACCERR is set, the CCIF flag cannot be cleared to launch a command. The ACCERR bit is cleared by writing a 1 to ACCERR. Writing a 0 to the ACCERR bit has no effect on ACCERR.<br>0 No access error detected<br>1 Access error detected |
| 4<br>FPVIOL | **Flash Protection Violation Flag** —The FPVIOL bit indicates an attempt was made to program or erase an address in a protected area of P-Flash memory during a command write sequence. The FPVIOL bit is cleared by writing a 1 to FPVIOL. Writing a 0 to the FPVIOL bit has no effect on FPVIOL. While FPVIOL is set, it is not possible to launch a command or start a command write sequence.<br>0 No protection violation detected<br>1 Protection violation detected |
| 3<br>MGBUSY | **Memory Controller Busy Flag** — The MGBUSY flag reflects the active state of the Memory Controller.<br>0 Memory Controller is idle<br>1 Memory Controller is busy executing a Flash command (CCIF = 0) or is handling internal EEE operations |
| 2<br>RSVD | **Reserved Bit** — This bit is reserved and always reads 0. |
| 1–0<br>MGSTAT[1:0] | **Memory Controller Command Completion Status Flag** — One or more MGSTAT flag bits are set if an error is detected during execution of a Flash command or during the Flash reset sequence. See Section 26.4.2, “Flash Command Description,” and Section 26.6, “Initialization” for details. |

### 26.3.2.8 Flash Error Status Register (FERSTAT)

The FERSTAT register reflects the error status of internal Flash operations.

Offset Module Base + 0x0007

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | ERSERIF | PGMERIF | 0 | EPVIOLIF | ERSVIF1 | ERSVIF0 | DFDIF | SFDIF |
| W | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented or Reserved

**Figure 26-12. Flash Error Status Register (FERSTAT)**

All flags in the FERSTAT register are readable and only writable to clear the flag.

**Table 26-18. FERSTAT Field Descriptions**

| Field | Description |
|---|---|
| 7<br>ERSERIF | **EEE Erase Error Interrupt Flag** — The setting of the ERSERIF flag occurs due to an error in a Flash erase command that resulted in the erase operation not being successful during EEE operations. The ERSERIF flag is cleared by writing a 1 to ERSERIF. Writing a 0 to the ERSERIF flag has no effect on ERSERIF. While ERSERIF is set, it is possible to write to the buffer RAM EEE partition but the data written will not be transferred to the D-Flash EEE partition.<br>0 Erase command successfully completed on the D-Flash EEE partition<br>1 Erase command failed on the D-Flash EEE partition |
| 6<br>PGMERIF | **EEE Program Error Interrupt Flag** — The setting of the PGMERIF flag occurs due to an error in a Flash program command that resulted in the program operation not being successful during EEE operations. The PGMERIF flag is cleared by writing a 1 to PGMERIF. Writing a 0 to the PGMERIF flag has no effect on PGMERIF. While PGMERIF is set, it is possible to write to the buffer RAM EEE partition but the data written will not be transferred to the D-Flash EEE partition.<br>0 Program command successfully completed on the D-Flash EEE partition<br>1 Program command failed on the D-Flash EEE partition |
| 4<br>EPVIOLIF | **EEE Protection Violation Interrupt Flag** —The setting of the EPVIOLIF flag indicates an attempt was made to write to a protected area of the buffer RAM EEE partition. The EPVIOLIF flag is cleared by writing a 1 to EPVIOLIF. Writing a 0 to the EPVIOLIF flag has no effect on EPVIOLIF. While EPVIOLIF is set, it is possible to write to the buffer RAM EEE partition as long as the address written to is not in a protected area.<br>0 No EEE protection violation<br>1 EEE protection violation detected |
| 3<br>ERSVIF1 | **EEE Error Interrupt 1 Flag** —The setting of the ERSVIF1 flag indicates that the memory controller was unable to change the state of a D-Flash EEE sector. The ERSVIF1 flag is cleared by writing a 1 to ERSVIF1. Writing a 0 to the ERSVIF1 flag has no effect on ERSVIF1. While ERSVIF1 is set, it is possible to write to the buffer RAM EEE partition but the data written will not be transferred to the D-Flash EEE partition.<br>0 No EEE sector state change error detected<br>1 EEE sector state change error detected |
| 2<br>ERSVIF0 | **EEE Error Interrupt 0 Flag** —The setting of the ERSVIF0 flag indicates that the memory controller was unable to format a D-Flash EEE sector for EEE use. The ERSVIF0 flag is cleared by writing a 1 to ERSVIF0. Writing a 0 to the ERSVIF0 flag has no effect on ERSVIF0. While ERSVIF0 is set, it is possible to write to the buffer RAM EEE partition but the data written will not be transferred to the D-Flash EEE partition.<br>0 No EEE sector format error detected<br>1 EEE sector format error detected |
| 1<br>DFDIF | **Double Bit Fault Detect Interrupt Flag** — The setting of the DFDIF flag indicates that a double bit fault was detected in the stored parity and data bits during a Flash array read operation or that a Flash array read operation was attempted on a Flash block that was under a Flash command operation. The DFDIF flag is cleared by writing a 1 to DFDIF. Writing a 0 to DFDIF has no effect on DFDIF.<br>0 No double bit fault detected<br>1 Double bit fault detected or an invalid Flash array read operation attempted |
| 0<br>SFDIF | **Single Bit Fault Detect Interrupt Flag** — With the IGNSF bit in the FCNFG register clear, the SFDIF flag indicates that a single bit fault was detected in the stored parity and data bits during a Flash array read operation or that a Flash array read operation was attempted on a Flash block that was under a Flash command operation. The SFDIF flag is cleared by writing a 1 to SFDIF. Writing a 0 to SFDIF has no effect on SFDIF.<br>0 No single bit fault detected<br>1 Single bit fault detected and corrected or an invalid Flash array read operation attempted |

### 26.3.2.9 P-Flash Protection Register (FPROT)

The FPROT register defines which P-Flash sectors are protected against program and erase operations.

Offset Module Base + 0x0008

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | FPOPEN | RNV6 | FPHDIS | FPHS[1:0] | | FPLDIS | FPLS[1:0] | |
| W | | | | | | | | |
| Reset | F | F | F | F | F | F | F | F |

= Unimplemented or Reserved

**Figure 26-13. Flash Protection Register (FPROT)**

The (unreserved) bits of the FPROT register are writable with the restriction that the size of the protected region can only be increased (see Section 26.3.2.9.1, “P-Flash Protection Restrictions,” and Table 26-23).

During the reset sequence, the FPROT register is loaded with the contents of the P-Flash protection byte in the Flash configuration field at global address 0x7F_FF0C located in P-Flash memory (see Table 26-3) as indicated by reset condition ‘F’ in Figure 26-13. To change the P-Flash protection that will be loaded during the reset sequence, the upper sector of the P-Flash memory must be unprotected, then the P-Flash protection byte must be reprogrammed. If a double bit fault is detected while reading the P-Flash phrase containing the P-Flash protection byte during the reset sequence, the FPOPEN bit will be cleared and remaining bits in the FPROT register will be set to leave the P-Flash memory fully protected.

Trying to alter data in any protected area in the P-Flash memory will result in a protection violation error and the FPVIOL bit will be set in the FSTAT register. The block erase of a P-Flash block is not possible if any of the P-Flash sectors contained in the same P-Flash block are protected.

**Table 26-19. FPROT Field Descriptions**

| Field | Description |
|---|---|
| 7<br>FPOPEN | **Flash Protection Operation Enable** — The FPOPEN bit determines the protection function for program or erase operations as shown in Table 26-20 for the P-Flash block.<br>0 When FPOPEN is clear, the FPHDIS and FPLDIS bits define unprotected address ranges as specified by the corresponding FPHS and FPLS bits<br>1 When FPOPEN is set, the FPHDIS and FPLDIS bits enable protection for the address range specified by the corresponding FPHS and FPLS bits |
| 6<br>RNV[6] | **Reserved Nonvolatile Bit** — The RNV bit should remain in the erased state for future enhancements. |
| 5<br>FPHDIS | **Flash Protection Higher Address Range Disable** — The FPHDIS bit determines whether there is a protected/unprotected area in a specific region of the P-Flash memory ending with global address 0x7F_FFFF.<br>0 Protection/Unprotection enabled<br>1 Protection/Unprotection disabled |
| 4–3<br>FPHS[1:0] | **Flash Protection Higher Address Size** — The FPHS bits determine the size of the protected/unprotected area in P-Flash memory as shown inTable 26-21. The FPHS bits can only be written to while the FPHDIS bit is set. |
| 2<br>FPLDIS | **Flash Protection Lower Address Range Disable** — The FPLDIS bit determines whether there is a protected/unprotected area in a specific region of the P-Flash memory beginning with global address 0x7F_8000.<br>0 Protection/Unprotection enabled<br>1 Protection/Unprotection disabled |
| 1–0<br>FPLS[1:0] | **Flash Protection Lower Address Size** — The FPLS bits determine the size of the protected/unprotected area in P-Flash memory as shown in Table 26-22. The FPLS bits can only be written to while the FPLDIS bit is set. |

**Table 26-20. P-Flash Protection Function**

| FPOPEN | FPHDIS | FPLDIS | Function[1] |
|---|---|---|---|
| 1 | 1 | 1 | No P-Flash Protection |
| 1 | 1 | 0 | Protected Low Range |
| 1 | 0 | 1 | Protected High Range |
| 1 | 0 | 0 | Protected High and Low Ranges |
| 0 | 1 | 1 | Full P-Flash Memory Protected |
| 0 | 1 | 0 | Unprotected Low Range |
| 0 | 0 | 1 | Unprotected High Range |
| 0 | 0 | 0 | Unprotected High and Low Ranges |

1. For range sizes, refer to Table 26-21 and Table 26-22.

**Table 26-21. P-Flash Protection Higher Address Range**

| FPHS[1:0] | Global Address Range | Protected Size |
|---|---|---|
| 00 | 0x7F_F800–0x7F_FFFF | 2 Kbytes |
| 01 | 0x7F_F000–0x7F_FFFF | 4 Kbytes |
| 10 | 0x7F_E000–0x7F_FFFF | 8 Kbytes |
| 11 | 0x7F_C000–0x7F_FFFF | 16 Kbytes |

**Table 26-22. P-Flash Protection Lower Address Range**

| FPLS[1:0] | Global Address Range | Protected Size |
|---|---|---|
| 00 | 0x7F_8000–0x7F_83FF | 1 Kbyte |
| 01 | 0x7F_8000–0x7F_87FF | 2 Kbytes |
| 10 | 0x7F_8000–0x7F_8FFF | 4 Kbytes |
| 11 | 0x7F_8000–0x7F_9FFF | 8 Kbytes |

All possible P-Flash protection scenarios are shown in Figure 26-14. Although the protection scheme is loaded from the Flash memory at global address 0x7F_FF0C during the reset sequence, it can be changed by the user. The P-Flash protection scheme can be used by applications requiring reprogramming in single chip mode while providing as much protection as possible if reprogramming is not required.

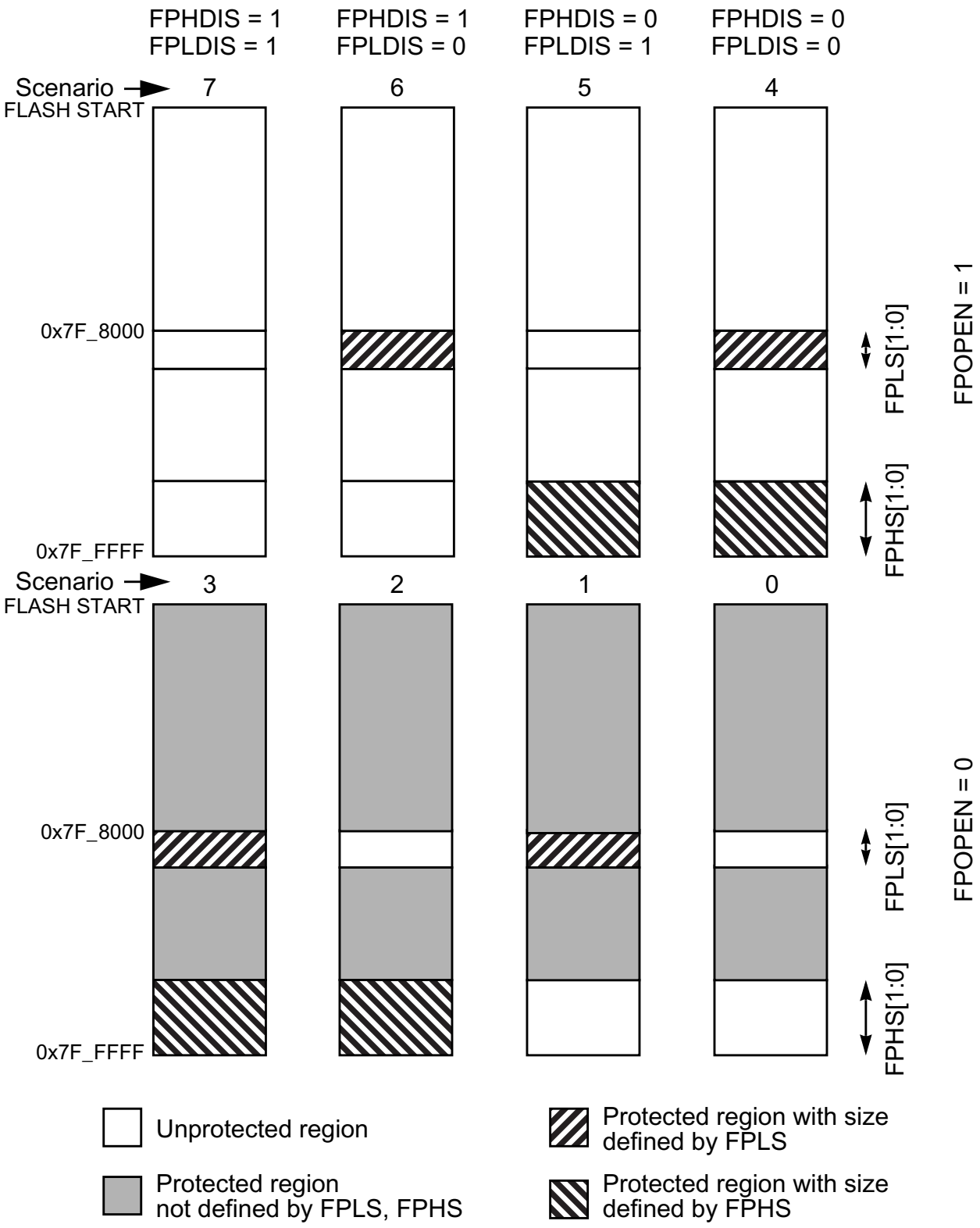



**Figure 26-14. P-Flash Protection Scenarios**

#### 26.3.2.9.1 P-Flash Protection Restrictions

The general guideline is that P-Flash protection can only be added and not removed. Table 26-23 specifies all valid transitions between P-Flash protection scenarios. Any attempt to write an invalid scenario to the FPROT register will be ignored. The contents of the FPROT register reflect the active protection scenario. See the FPHS and FPLS bit descriptions for additional restrictions.

**Table 26-23. P-Flash Protection Scenario Transitions**

| From Protection Scenario | To Protection Scenario[1] | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| 0 | X | X | X | X | | | | |
| 1 | | X | | X | | | | |
| 2 | | | X | X | | | | |
| 3 | | | | X | | | | |
| 4 | | | | X | X | | | |
| 5 | | | X | X | X | X | | |
| 6 | | X | | X | X | | X | |
| 7 | X | X | X | X | X | X | X | X |

1. Allowed transitions marked with X, see Figure 26-14 for a definition of the scenarios.

### 26.3.2.10 EEE Protection Register (EPROT)

The EPROT register defines which buffer RAM EEE partition areas are protected against writes.

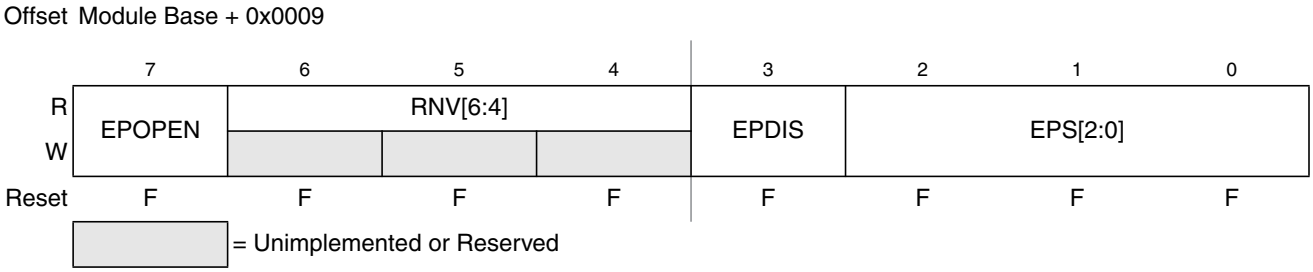



**Figure 26-15. EEE Protection Register (EPROT)**

All bits in the EPROT register are readable and writable except for RNV[6:4] which are only readable. The EPOPEN and EPDIS bits can only be written to the protected state. The EPS bits can be written anytime until the EPDIS bit is cleared. If the EPOPEN bit is cleared, the state of the EPDIS and EPS bits is irrelevant.

During the reset sequence, the EPROT register is loaded from the EEE protection byte in the Flash configuration field at global address 0x7F_FF0D located in P-Flash memory (see Table 26-3) as indicated by reset condition F in Figure 26-15. To change the EEE protection that will be loaded during the reset sequence, the P-Flash sector containing the EEE protection byte must be unprotected, then the EEE protection byte must be programmed. If a double bit fault is detected while reading the P-Flash phrase

containing the EEE protection byte during the reset sequence, the EPOPEN bit will be cleared and remaining bits in the EPROT register will be set to leave the buffer RAM EEE partition fully protected.

Trying to write data to any protected area in the buffer RAM EEE partition will result in a protection violation error and the EPVIOLIF flag will be set in the FERSTAT register. Trying to write data to any protected area in the buffer RAM partitioned for user access will not be prevented and the EPVIOLIF flag in the FERSTAT register will not set.

**Table 26-24. EPROT Field Descriptions**

| Field | Description |
|---|---|
| 7<br>EPOPEN | **Enables writes to the Buffer RAM partitioned for EEE**<br>0 The entire buffer RAM EEE partition is protected from writes<br>1 Unprotected buffer RAM EEE partition areas are enabled for writes |
| 6–4<br>RNV[6:4] | **Reserved Nonvolatile Bits** — The RNV bits should remain in the erased state for future enhancements |
| 3<br>EPDIS | **Buffer RAM Protection Address Range Disable** — The EPDIS bit determines whether there is a protected area in a specific region of the buffer RAM EEE partition.<br>0 Protection enabled<br>1 Protection disabled |
| 2–0<br>EPS[2:0] | **Buffer RAM Protection Size** — The EPS[2:0] bits determine the size of the protected area in the buffer RAM EEE partition as shown inTable 26-21. The EPS bits can only be written to while the EPDIS bit is set. |

**Table 26-25. Buffer RAM EEE Partition Protection Address Range**

| EPS[2:0] | Global Address Range | Protected Size |
|---|---|---|
| 000 | 0x13_FFC0 – 0x13_FFFF | 64 bytes |
| 001 | 0x13_FF80 – 0x13_FFFF | 128 bytes |
| 010 | 0x13_FF40 – 0x13_FFFF | 192 bytes |
| 011 | 0x13_FF00 – 0x13_FFFF | 256 bytes |
| 100 | 0x13_FEC0 – 0x13_FFFF | 320 bytes |
| 101 | 0x13_FE80 – 0x13_FFFF | 384 bytes |
| 110 | 0x13_FE40 – 0x13_FFFF | 448 bytes |
| 111 | 0x13_FE00 – 0x13_FFFF | 512 bytes |

### 26.3.2.11 Flash Common Command Object Register (FCCOB)

The FCCOB is an array of six words addressed via the CCOBIX index found in the FCCOBIX register. Byte wide reads and writes are allowed to the FCCOB register.

Offset Module Base + 0x000A

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | CCOB[15:8] | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 26-16. Flash Common Command Object High Register (FCCOBHI)**

Offset Module Base + 0x000B

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | CCOB[7:0] | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 26-17. Flash Common Command Object Low Register (FCCOBLO)**

### 26.3.2.11.1 FCCOB - NVM Command Mode

NVM command mode uses the indexed FCCOB register to provide a command code and its relevant parameters to the Memory Controller. The user first sets up all required FCCOB fields and then initiates the command's execution by writing a 1 to the CCIF bit in the FSTAT register (a 1 written by the user clears the CCIF command completion flag to 0). When the user clears the CCIF bit in the FSTAT register all FCCOB parameter fields are locked and cannot be changed by the user until the command completes (as evidenced by the Memory Controller returning CCIF to 1). Some commands return information to the FCCOB register array.

The generic format for the FCCOB parameter fields in NVM command mode is shown in Table 26-26. The return values are available for reading after the CCIF flag in the FSTAT register has been returned to 1 by the Memory Controller. Writes to the unimplemented parameter fields (CCOBIX = 110 and CCOBIX = 111) are ignored with reads from these fields returning 0x0000.

Table 26-26 shows the generic Flash command format. The high byte of the first word in the CCOB array contains the command code, followed by the parameters for this specific Flash command. For details on the FCCOB settings required by each command, see the Flash command descriptions in Section 26.4.2.

**Table 26-26. FCCOB - NVM Command Mode (Typical Usage)**

| CCOBIX[2:0] | Byte | FCCOB Parameter Fields (NVM Command Mode) |
|---|---|---|
| 000 | HI | FCMD[7:0] defining Flash command |
| | LO | 0, Global address [22:16] |
| 001 | HI | Global address [15:8] |
| | LO | Global address [7:0] |
| 010 | HI | Data 0 [15:8] |
| | LO | Data 0 [7:0] |

**Table 26-26. FCCOB - NVM Command Mode (Typical Usage)**

| CCOBIX[2:0] | Byte | FCCOB Parameter Fields (NVM Command Mode) |
|---|---|---|
| 011 | HI | Data 1 [15:8] |
| | LO | Data 1 [7:0] |
| 100 | HI | Data 2 [15:8] |
| | LO | Data 2 [7:0] |
| 101 | HI | Data 3 [15:8] |
| | LO | Data 3 [7:0] |

### 26.3.2.12 EEE Tag Counter Register (ETAG)

The ETAG register contains the number of outstanding words in the buffer RAM EEE partition that need to be programmed into the D-Flash EEE partition. The ETAG register is decremented prior to the related tagged word being programmed into the D-Flash EEE partition. All tagged words have been programmed into the D-Flash EEE partition once all bits in the ETAG register read 0 and the MGBUSY flag in the FSTAT register reads 0.

Offset Module Base + 0x000C

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | ETAG[15:8] | | | | | | | |
| W | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented or Reserved

**Figure 26-18. EEE Tag Counter High Register (ETAGHI)**

Offset Module Base + 0x000D

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | ETAG[7:0] | | | | | | | |
| W | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented or Reserved

**Figure 26-19. EEE Tag Counter Low Register (ETAGLO)**

All ETAG bits are readable but not writable and are cleared by the Memory Controller.

### 26.3.2.13 Flash ECC Error Results Register (FECCR)

The FECCR registers contain the result of a detected ECC fault for both single bit and double bit faults. The FECCR register provides access to several ECC related fields as defined by the ECCRIX index bits in the FECCRIX register (see Section 26.3.2.4). Once ECC fault information has been stored, no other

fault information will be recorded until the specific ECC fault flag has been cleared. In the event of simultaneous ECC faults, the priority for fault recording is:

1. Double bit fault over single bit fault
2. CPU over XGATE

Offset Module Base + 0x000E

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | ECCR[15:8] | | | | | | | |
| W | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented or Reserved

**Figure 26-20. Flash ECC Error Results High Register (FECCRHI)**

Offset Module Base + 0x000F

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | ECCR[7:0] | | | | | | | |
| W | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented or Reserved

**Figure 26-21. Flash ECC Error Results Low Register (FECCRLO)**

All FECCR bits are readable but not writable.

**Table 26-27. FECCR Index Settings**

| ECCRIX[2:0] | FECCR Register Content | | |
|---|---|---|---|
| | **Bits [15:8]** | **Bit[7]** | **Bits[6:0]** |
| 000 | Parity bits read from Flash block | CPU or XGATE source identity | Global address [22:16] |
| 001 | Global address [15:0] | | |
| 010 | Data 0 [15:0] | | |
| 011 | Data 1 [15:0] (P-Flash only) | | |
| 100 | Data 2 [15:0] (P-Flash only) | | |
| 101 | Data 3 [15:0] (P-Flash only) | | |
| 110 | Not used, returns 0x0000 when read | | |
| 111 | Not used, returns 0x0000 when read | | |

**Table 26-28. FECCR Index=000 Bit Descriptions**

| Field | Description |
|---|---|
| 15:8<br>PAR[7:0] | **ECC Parity Bits** — Contains the 8 parity bits from the 72 bit wide P-Flash data word or the 6 parity bits, allocated to PAR[5:0], from the 22 bit wide D-Flash word with PAR[7:6]=00. |
| 7<br>XBUS01 | **Bus Source Identifier** — The XBUS01 bit determines whether the ECC error was caused by a read access from the CPU or XGATE.<br>0 ECC Error happened on the CPU access<br>1 ECC Error happened on the XGATE access |
| 6–0<br>GADDR[22:16] | **Global Address** — The GADDR[22:16] field contains the upper seven bits of the global address having caused the error. |

The P-Flash word addressed by ECCRIX = 001 contains the lower 16 bits of the global address. The following four words addressed by ECCRIX = 010 to 101 contain the 64-bit wide data phrase. The four data words and the parity byte are the uncorrected data read from the P-Flash block.

The D-Flash word addressed by ECCRIX = 001 contains the lower 16 bits of the global address. The uncorrected 16-bit data word is addressed by ECCRIX = 010.

### 26.3.2.14 Flash Option Register (FOPT)

The FOPT register is the Flash option register.

Offset Module Base + 0x0010

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | NV[7:0] | | | | | | | |
| W | | | | | | | | |
| Reset | F | F | F | F | F | F | F | F |

= Unimplemented or Reserved

**Figure 26-22. Flash Option Register (FOPT)**

All bits in the FOPT register are readable but are not writable.

During the reset sequence, the FOPT register is loaded from the Flash nonvolatile byte in the Flash configuration field at global address 0x7F_FF0E located in P-Flash memory (see Table 26-3) as indicated by reset condition F in Figure 26-22. If a double bit fault is detected while reading the P-Flash phrase containing the Flash nonvolatile byte during the reset sequence, all bits in the FOPT register will be set.

**Table 26-29. FOPT Field Descriptions**

| Field | Description |
|---|---|
| 7–0<br>NV[7:0] | **Nonvolatile Bits** — The NV[7:0] bits are available as nonvolatile bits. Refer to the device user guide for proper use of the NV bits. |

### 26.3.2.15 Flash Reserved0 Register (FRSV0)

This Flash register is reserved for factory testing.

Offset Module Base + 0x0011

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| W | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented or Reserved

**Figure 26-23. Flash Reserved0 Register (FRSV0)**

All bits in the FRSV0 register read 0 and are not writable.

### 26.3.2.16 Flash Reserved1 Register (FRSV1)

This Flash register is reserved for factory testing.

Offset Module Base + 0x0012

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| W | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented or Reserved

**Figure 26-24. Flash Reserved1 Register (FRSV1)**

All bits in the FRSV1 register read 0 and are not writable.

### 26.3.2.17 Flash Reserved2 Register (FRSV2)

This Flash register is reserved for factory testing.

Offset Module Base + 0x0013

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| W | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented or Reserved

**Figure 26-25. Flash Reserved2 Register (FRSV2)**

All bits in the FRSV2 register read 0 and are not writable.

# 26.4 Functional Description

## 26.4.1 Flash Command Operations

Flash command operations are used to modify Flash memory contents or configure module resources for EEE operation.

The next sections describe:

- How to write the FCLKDIV register that is used to generate a time base (FCLK) derived from OSCCLK for Flash program and erase command operations
- The command write sequence used to set Flash command parameters and launch execution
- Valid Flash commands available for execution

### 26.4.1.1 Writing the FCLKDIV Register

Prior to issuing any Flash program or erase command after a reset, the user is required to write the FCLKDIV register to divide OSCCLK down to a target FCLK of 1 MHz. Table 26-9 shows recommended values for the FDIV field based on OSCCLK frequency.

**NOTE**

Programming or erasing the Flash memory cannot be performed if the bus clock runs at less than 1 MHz. Setting FDIV too high can destroy the Flash memory due to overstress. Setting FDIV too low can result in incomplete programming or erasure of the Flash memory cells.

When the FCLKDIV register is written, the FDIVLD bit is set automatically. If the FDIVLD bit is 0, the FCLKDIV register has not been written since the last reset. If the FCLKDIV register has not been written, any Flash program or erase command loaded during a command write sequence will not execute and the ACCERR bit in the FSTAT register will set.

### 26.4.1.2 Command Write Sequence

The Memory Controller will launch all valid Flash commands entered using a command write sequence.

Before launching a command, the ACCERR and FPVIOL bits in the FSTAT register must be clear (see Section 26.3.2.7) and the CCIF flag should be tested to determine the status of the current command write sequence. If CCIF is 0, the previous command write sequence is still active, a new command write sequence cannot be started, and all writes to the FCCOB register are ignored.

**CAUTION**

Writes to any Flash register must be avoided while a Flash command is active (CCIF=0) to prevent corruption of Flash register contents and Memory Controller behavior.

#### 26.4.1.2.1 Define FCCOB Contents

The FCCOB parameter fields must be loaded with all required parameters for the Flash command being executed. Access to the FCCOB parameter fields is controlled via the CCOBIX bits in the FCCOBIX register (see Section 26.3.2.3).

The contents of the FCCOB parameter fields are transferred to the Memory Controller when the user clears the CCIF command completion flag in the FSTAT register (writing 1 clears the CCIF to 0). The CCIF flag will remain clear until the Flash command has completed. Upon completion, the Memory Controller will return CCIF to 1 and the FCCOB register will be used to communicate any results. The flow for a generic command write sequence is shown in Figure 26-26.

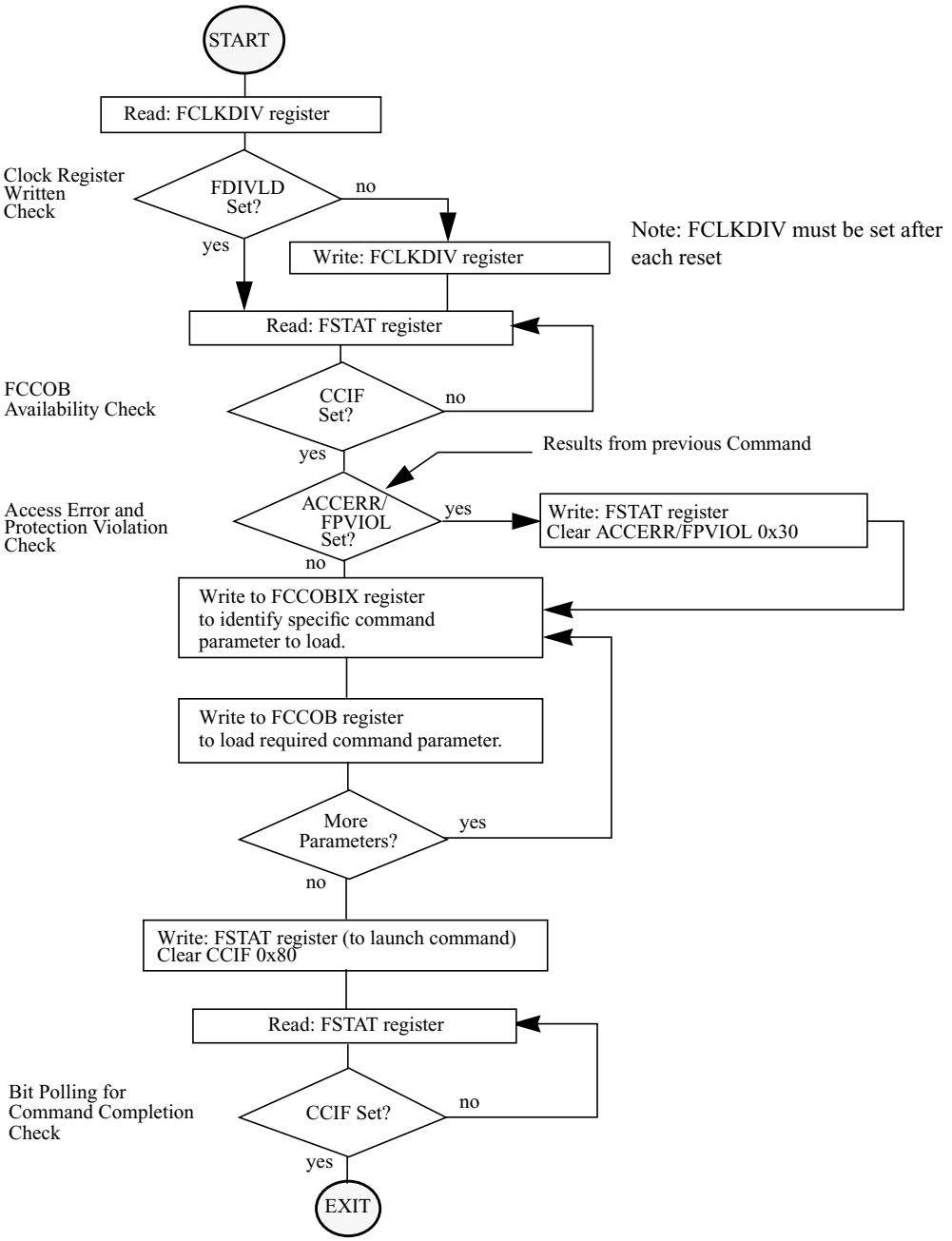



**Figure 26-26. Generic Flash Command Write Sequence Flowchart**

### 26.4.1.3 Valid Flash Module Commands

**Table 26-30. Flash Commands by Mode**

| FCMD | Command | Unsecured | | | | Secured | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | NS (1) | NX (2) | SS(3) | ST(4) | NS (5) | NX (6) | SS(7) | ST(8) |
| 0x01 | Erase Verify All Blocks | * | * | * | * | * | * | * | * |
| 0x02 | Erase Verify Block | * | * | * | * | * | * | * | * |
| 0x03 | Erase Verify P-Flash Section | * | * | * | * | * | | | |
| 0x04 | Read Once | * | * | * | * | * | | | |
| 0x05 | Load Data Field | * | * | * | * | * | | | |
| 0x06 | Program P-Flash | * | * | * | * | * | | | |
| 0x07 | Program Once | * | * | * | * | * | | | |
| 0x08 | Erase All Blocks | | | * | * | | | * | * |
| 0x09 | Erase P-Flash Block | * | * | * | * | * | | | |
| 0x0A | Erase P-Flash Sector | * | * | * | * | * | | | |
| 0x0B | Unsecure Flash | | | * | * | | | * | * |
| 0x0C | Verify Backdoor Access Key | * | | | | * | | | |
| 0x0D | Set User Margin Level | * | * | * | * | * | | | |
| 0x0E | Set Field Margin Level | | | * | * | | | | |
| 0x0F | Full Partition D-Flash | | | * | * | | | | |
| 0x10 | Erase Verify D-Flash Section | * | * | * | * | * | | | |
| 0x11 | Program D-Flash | * | * | * | * | * | | | |
| 0x12 | Erase D-Flash Sector | * | * | * | * | * | | | |
| 0x13 | Enable EEPROM Emulation | * | * | * | * | * | * | * | * |
| 0x14 | Disable EEPROM Emulation | * | * | * | * | * | * | * | * |
| 0x15 | EEPROM Emulation Query | * | * | * | * | * | * | * | * |
| 0x20 | Partition D-Flash | * | * | * | * | * | * | * | * |

1. Unsecured Normal Single Chip mode.
2. Unsecured Normal Expanded mode.
3. Unsecured Special Single Chip mode.
4. Unsecured Special Mode.
5. Secured Normal Single Chip mode.
6. Secured Normal Expanded mode.
7. Secured Special Single Chip mode.
8. Secured Special Mode.

### 26.4.1.4 P-Flash Commands

Table 26-31 summarizes the valid P-Flash commands along with the effects of the commands on the P-Flash block and other resources within the Flash module.

**Table 26-31. P-Flash Commands**

| FCMD | Command | Function on P-Flash Memory |
|---|---|---|
| 0x01 | Erase Verify All Blocks | Verify that all P-Flash (and D-Flash) blocks are erased. |
| 0x02 | Erase Verify Block | Verify that a P-Flash block is erased. |
| 0x03 | Erase Verify P-Flash Section | Verify that a given number of words starting at the address provided are erased. |
| 0x04 | Read Once | Read a dedicated 64 byte field in the nonvolatile information register in P-Flash block 0 that was previously programmed using the Program Once command. |
| 0x05 | Load Data Field | Load data for simultaneous multiple P-Flash block operations. |
| 0x06 | Program P-Flash | Program a phrase in a P-Flash block and any previously loaded phrases for any other P-Flash block (see Load Data Field command). |
| 0x07 | Program Once | Program a dedicated 64 byte field in the nonvolatile information register in P-Flash block 0 that is allowed to be programmed only once. |
| 0x08 | Erase All Blocks | Erase all P-Flash (and D-Flash) blocks.<br>An erase of all Flash blocks is only possible when the FPLDIS, FPHDIS, and FPOPEN bits in the FPROT register and the EPDIS and EPOPEN bits in the EPROT register are set prior to launching the command. |
| 0x09 | Erase P-Flash Block | Erase a single P-Flash block.<br>An erase of the full P-Flash block is only possible when FPLDIS, FPHDIS and FPOPEN bits in the FPROT register are set prior to launching the command. |
| 0x0A | Erase P-Flash Sector | Erase all bytes in a P-Flash sector. |
| 0x0B | Unsecure Flash | Supports a method of releasing MCU security by erasing all P-Flash (and D-Flash) blocks and verifying that all P-Flash (and D-Flash) blocks are erased. |
| 0x0C | Verify Backdoor Access Key | Supports a method of releasing MCU security by verifying a set of security keys. |
| 0x0D | Set User Margin Level | Specifies a user margin read level for all P-Flash blocks. |
| 0x0E | Set Field Margin Level | Specifies a field margin read level for all P-Flash blocks (special modes only). |

### 26.4.1.5 D-Flash and EEE Commands

Table 26-32 summarizes the valid D-Flash and EEE commands along with the effects of the commands on the D-Flash block and EEE operation.

**Table 26-32. D-Flash Commands**

| FCMD | Command | Function on D-Flash Memory |
|---|---|---|
| 0x01 | Erase Verify All Blocks | Verify that all D-Flash (and P-Flash) blocks are erased. |
| 0x02 | Erase Verify Block | Verify that the D-Flash block is erased. |

**Table 26-32. D-Flash Commands**

| FCMD | Command | Function on D-Flash Memory |
|---|---|---|
| 0x08 | Erase All Blocks | Erase all D-Flash (and P-Flash) blocks.<br>An erase of all Flash blocks is only possible when the FPLDIS, FPHDIS, and FPOPEN bits in the FPROT register and the EPDIS and EPOPEN bits in the EPROT register are set prior to launching the command. |
| 0x0B | Unsecure Flash | Supports a method of releasing MCU security by erasing all D-Flash (and P-Flash) blocks and verifying that all D-Flash (and P-Flash) blocks are erased. |
| 0x0D | Set User Margin Level | Specifies a user margin read level for the D-Flash block. |
| 0x0E | Set Field Margin Level | Specifies a field margin read level for the D-Flash block (special modes only). |
| 0x0F | Full Partition D-Flash | Erase the D-Flash block and partition an area of the D-Flash block for user access. |
| 0x10 | Erase Verify D-Flash Section | Verify that a given number of words starting at the address provided are erased. |
| 0x11 | Program D-Flash | Program up to four words in the D-Flash block. |
| 0x12 | Erase D-Flash Sector | Erase all bytes in a sector of the D-Flash block. |
| 0x13 | Enable EEPROM Emulation | Enable EEPROM emulation where writes to the buffer RAM EEE partition will be copied to the D-Flash EEE partition. |
| 0x14 | Disable EEPROM Emulation | Suspend all current erase and program activity related to EEPROM emulation but leave current EEE tags set. |
| 0x15 | EEPROM Emulation Query | Returns EEE partition and status variables. |
| 0x20 | Partition D-Flash | Partition an area of the D-Flash block for user access. |

## 26.4.2 Flash Command Description

This section provides details of all available Flash commands launched by a command write sequence. The ACCERR bit in the FSTAT register will be set during the command write sequence if any of the following illegal steps are performed, causing the command not to be processed by the Memory Controller:

- Starting any command write sequence that programs or erases Flash memory before initializing the FCLKDIV register
- Writing an invalid command as part of the command write sequence
- For additional possible errors, refer to the error handling table provided for each command

If a Flash block is read during execution of an algorithm (CCIF = 0) on that same block, the read operation will return invalid data. If the SFDIF or DFDIF flags were not previously set when the invalid read operation occurred, both the SFDIF and DFDIF flags will be set and the FECCR registers will be loaded with the global address used in the invalid read operation with the data and parity fields set to all 0.

If the ACCERR or FPVIOL bits are set in the FSTAT register, the user must clear these bits before starting any command write sequence (see Section 26.3.2.7).

**CAUTION**

A Flash word or phrase must be in the erased state before being programmed. Cumulative programming of bits within a Flash word or phrase is not allowed.

### 26.4.2.1 Erase Verify All Blocks Command

The Erase Verify All Blocks command will verify that all P-Flash and D-Flash blocks have been erased.

**Table 26-33. Erase Verify All Blocks Command FCCOB Requirements**

| CCOBIX[2:0] | FCCOB Parameters | |
|---|---|---|
| 000 | 0x01 | Not required |

Upon clearing CCIF to launch the Erase Verify All Blocks command, the Memory Controller will verify that the entire Flash memory space is erased. The CCIF flag will set after the Erase Verify All Blocks operation has completed.

**Table 26-34. Erase Verify All Blocks Command Error Handling**

| Register | Error Bit | Error Condition |
|---|---|---|
| FSTAT | ACCERR | Set if CCOBIX[2:0] != 000 at command launch |
| | | Set if a Load Data Field command sequence is currently active |
| | FPVIOL | None |
| | MGSTAT1 | Set if any errors have been encountered during the read(1) |
| | MGSTAT0 | Set if any non-correctable errors have been encountered during the read[1] |
| FERSTAT | EPVIOLIF | None |

1. As found in the memory map for FTM512K3.

### 26.4.2.2 Erase Verify Block Command

The Erase Verify Block command allows the user to verify that an entire P-Flash or D-Flash block has been erased. The FCCOB upper global address bits determine which block must be verified.

**Table 26-35. Erase Verify Block Command FCCOB Requirements**

| CCOBIX[2:0] | FCCOB Parameters | |
|---|---|---|
| 000 | 0x02 | Global address [22:16] of the Flash block to be verified. |

Upon clearing CCIF to launch the Erase Verify Block command, the Memory Controller will verify that the selected P-Flash or D-Flash block is erased. The CCIF flag will set after the Erase Verify Block operation has completed.

**Table 26-36. Erase Verify Block Command Error Handling**

| Register | Error Bit | Error Condition |
|---|---|---|
| FSTAT | ACCERR | Set if CCOBIX[2:0] != 000 at command launch |
| | | Set if a Load Data Field command sequence is currently active |
| | | Set if an invalid global address [22:16] is supplied(1) |
| | FPVIOL | None |
| | MGSTAT1 | Set if any errors have been encountered during the read(2) |
| | MGSTAT0 | Set if any non-correctable errors have been encountered during the read[2] |
| FERSTAT | EPVIOLIF | None |

1. As defined by the memory map for FTM512K3.
2. As found in the memory map for FTM512K3.

### 26.4.2.3 Erase Verify P-Flash Section Command

The Erase Verify P-Flash Section command will verify that a section of code in the P-Flash memory is erased. The Erase Verify P-Flash Section command defines the starting point of the code to be verified and the number of phrases. The section to be verified cannot cross a 256 Kbyte boundary in the P-Flash memory space.

**Table 26-37. Erase Verify P-Flash Section Command FCCOB Requirements**

| CCOBIX[2:0] | FCCOB Parameters | |
|---|---|---|
| 000 | 0x03 | Global address [22:16] of a P-Flash block |
| 001 | Global address [15:0] of the first phrase to be verified | |
| 010 | Number of phrases to be verified | |

Upon clearing CCIF to launch the Erase Verify P-Flash Section command, the Memory Controller will verify the selected section of Flash memory is erased. The CCIF flag will set after the Erase Verify P-Flash Section operation has completed.

**Table 26-38. Erase Verify P-Flash Section Command Error Handling**

| Register | Error Bit | Error Condition |
|---|---|---|
| FSTAT | ACCERR | Set if CCOBIX[2:0] != 010 at command launch |
| | | Set if a Load Data Field command sequence is currently active |
| | | Set if command not available in current mode (see Table 26-30) |
| | | Set if an invalid global address [22:0] is supplied(1) |
| | | Set if a misaligned phrase address is supplied (global address [2:0] != 000) |
| | | Set if the requested section crosses a 256 Kbyte boundary |
| | FPVIOL | None |
| | MGSTAT1 | Set if any errors have been encountered during the read(2) |
| | MGSTAT0 | Set if any non-correctable errors have been encountered during the read2 |
| FERSTAT | EPVIOLIF | None |

1. As defined by the memory map for FTM512K3.
2. As found in the memory map for FTM512K3.

### 26.4.2.4 Read Once Command

The Read Once command provides read access to a reserved 64 byte field (8 phrases) located in the nonvolatile information register of P-Flash block 0. The Read Once field is programmed using the Program Once command described in Section 26.4.2.7. The Read Once command must not be executed from the Flash block containing the Program Once reserved field to avoid code runaway.

**Table 26-39. Read Once Command FCCOB Requirements**

| CCOBIX[2:0] | FCCOB Parameters | |
|---|---|---|
| 000 | 0x04 | Not Required |
| 001 | Read Once phrase index (0x0000 - 0x0007) | |
| 010 | Read Once word 0 value | |
| 011 | Read Once word 1 value | |
| 100 | Read Once word 2 value | |
| 101 | Read Once word 3 value | |

Upon clearing CCIF to launch the Read Once command, a Read Once phrase is fetched and stored in the FCCOB indexed register. The CCIF flag will set after the Read Once operation has completed. Valid phrase index values for the Read Once command range from 0x0000 to 0x0007. During execution of the Read Once command, any attempt to read addresses within P-Flash block 0 will return invalid data.

**Table 26-40. Read Once Command Error Handling**

| Register | Error Bit | Error Condition |
|---|---|---|
| FSTAT | ACCERR | Set if CCOBIX[2:0] != 001 at command launch |
| | | Set if a Load Data Field command sequence is currently active |
| | | Set if command not available in current mode (see Table 26-30) |
| | | Set if an invalid phrase index is supplied |
| | FPVIOL | None |
| | MGSTAT1 | Set if any errors have been encountered during the read |
| | MGSTAT0 | Set if any non-correctable errors have been encountered during the read |
| FERSTAT | EPVIOLIF | None |

### 26.4.2.5 Load Data Field Command

The Load Data Field command is executed to provide FCCOB parameters for multiple P-Flash blocks for a future simultaneous program operation in the P-Flash memory space.

**Table 26-41. Load Data Field Command FCCOB Requirements**

| CCOBIX[2:0] | FCCOB Parameters | |
|---|---|---|
| 000 | 0x05 | Global address [22:16] to identify P-Flash block |
| 001 | Global address [15:0] of phrase location to be programmed(1) | |
| 010 | Word 0 | |
| 011 | Word 1 | |
| 100 | Word 2 | |
| 101 | Word 3 | |

1. Global address [2:0] must be 000

Upon clearing CCIF to launch the Load Data Field command, the FCCOB registers will be transferred to the Memory Controller and be programmed in the block specified at the global address given with a future Program P-Flash command launched on a P-Flash block. The CCIF flag will set after the Load Data Field operation has completed. Note that once a Load Data Field command sequence has been initiated, the Load Data Field command sequence will be cancelled if any command other than Load Data Field or the future Program P-Flash is launched. Similarly, if an error occurs after launching a Load Data Field or Program P-Flash command, the associated Load Data Field command sequence will be cancelled.

**Table 26-42. Load Data Field Command Error Handling**

| Register | Error Bit | Error Condition |
|---|---|---|
| FSTAT | ACCERR | Set if CCOBIX[2:0] != 101 at command launch |
| | | Set if command not available in current mode (see Table 26-30) |
| | | Set if an invalid global address [22:0] is supplied(1) |
| | | Set if a misaligned phrase address is supplied (global address [2:0] != 000) |
| | | Set if a Load Data Field command sequence is currently active and the selected block has previously been selected in the same command sequence |
| | | Set if a Load Data Field command sequence is currently active and global address [17:0] does not match that previously supplied in the same command sequence |
| | FPVIOL | Set if the global address [22:0] points to a protected area |
| | MGSTAT1 | None |
| | MGSTAT0 | None |
| FERSTAT | EPVIOLIF | None |

1. As defined by the memory map for FTM512K3.

### 26.4.2.6 Program P-Flash Command

The Program P-Flash operation will program a previously erased phrase in the P-Flash memory using an embedded algorithm.

**CAUTION**

A P-Flash phrase must be in the erased state before being programmed. Cumulative programming of bits within a Flash phrase is not allowed.

**Table 26-43. Program P-Flash Command FCCOB Requirements**

| CCOBIX[2:0] | FCCOB Parameters | |
|---|---|---|
| 000 | 0x06 | Global address [22:16] to identify P-Flash block |
| 001 | Global address [15:0] of phrase location to be programmed(1) | |
| 010 | Word 0 program value | |
| 011 | Word 1 program value | |
| 100 | Word 2 program value | |
| 101 | Word 3 program value | |

1. Global address [2:0] must be 000

Upon clearing CCIF to launch the Program P-Flash command, the Memory Controller will program the data words to the supplied global address and will then proceed to verify the data words read back as expected. The CCIF flag will set after the Program P-Flash operation has completed.

**Table 26-44. Program P-Flash Command Error Handling**

| Register | Error Bit | Error Condition |
|---|---|---|
| FSTAT | ACCERR | Set if CCOBIX[2:0] != 101 at command launch |
| | | Set if command not available in current mode (see Table 26-30) |
| | | Set if an invalid global address [22:0] is supplied(1) |
| | | Set if a misaligned phrase address is supplied (global address [2:0] != 000) |
| | | Set if a Load Data Field command sequence is currently active and the selected block has previously been selected in the same command sequence |
| | | Set if a Load Data Field command sequence is currently active and global address [17:0] does not match that previously supplied in the same command sequence |
| | FPVIOL | Set if the global address [22:0] points to a protected area |
| | MGSTAT1 | Set if any errors have been encountered during the verify operation |
| | MGSTAT0 | Set if any non-correctable errors have been encountered during the verify operation |
| FERSTAT | EPVIOLIF | None |

1. As defined by the memory map for FTM512K3.

### 26.4.2.7 Program Once Command

The Program Once command restricts programming to a reserved 64 byte field (8 phrases) in the nonvolatile information register located in P-Flash block 0. The Program Once reserved field can be read using the Read Once command as described in Section 26.4.2.4. The Program Once command must only be issued once since the nonvolatile information register in P-Flash block 0 cannot be erased. The Program Once command must not be executed from the Flash block containing the Program Once reserved field to avoid code runaway.

**Table 26-45. Program Once Command FCCOB Requirements**

| CCOBIX[2:0] | FCCOB Parameters | |
|---|---|---|
| 000 | 0x07 | Not Required |
| 001 | Program Once phrase index (0x0000 - 0x0007) | |
| 010 | Program Once word 0 value | |
| 011 | Program Once word 1 value | |
| 100 | Program Once word 2 value | |
| 101 | Program Once word 3 value | |

Upon clearing CCIF to launch the Program Once command, the Memory Controller first verifies that the selected phrase is erased. If erased, then the selected phrase will be programmed and then verified with read back. The CCIF flag will remain clear, setting only after the Program Once operation has completed.

The reserved nonvolatile information register accessed by the Program Once command cannot be erased and any attempt to program one of these phrases a second time will not be allowed. Valid phrase index

values for the Program Once command range from 0x0000 to 0x0007. During execution of the Program Once command, any attempt to read addresses within P-Flash block 0 will return invalid data.

**Table 26-46. Program Once Command Error Handling**

| Register | Error Bit | Error Condition |
|---|---|---|
| FSTAT | ACCERR | Set if CCOBIX[2:0] != 101 at command launch |
| | | Set if a Load Data Field command sequence is currently active |
| | | Set if command not available in current mode (see Table 26-30) |
| | | Set if an invalid phrase index is supplied |
| | | Set if the requested phrase has already been programmed(1) |
| | FPVIOL | None |
| | MGSTAT1 | Set if any errors have been encountered during the verify operation |
| | MGSTAT0 | Set if any non-correctable errors have been encountered during the verify operation |
| FERSTAT | EPVIOLIF | None |

1. If a Program Once phrase is initially programmed to 0xFFFF_FFFF_FFFF_FFFF, the Program Once command will be allowed to execute again on that same phrase.

### 26.4.2.8 Erase All Blocks Command

The Erase All Blocks operation will erase the entire P-Flash and D-Flash memory space including the EEE nonvolatile information register.

**Table 26-47. Erase All Blocks Command FCCOB Requirements**

| CCOBIX[2:0] | FCCOB Parameters | |
|---|---|---|
| 000 | 0x08 | Not required |

Upon clearing CCIF to launch the Erase All Blocks command, the Memory Controller will erase the entire Flash memory space and verify that it is erased. If the Memory Controller verifies that the entire Flash memory space was properly erased, security will be released. During the execution of this command (CCIF=0) the user must not write to any Flash module register. The CCIF flag will set after the Erase All Blocks operation has completed.

**Table 26-48. Erase All Blocks Command Error Handling**

| Register | Error Bit | Error Condition |
|---|---|---|
| FSTAT | ACCERR | Set if CCOBIX[2:0] != 000 at command launch |
| | | Set if a Load Data Field command sequence is currently active |
| | | Set if command not available in current mode (see Table 26-30) |
| | FPVIOL | Set if any area of the P-Flash memory is protected |
| | MGSTAT1 | Set if any errors have been encountered during the verify operation(1) |
| | MGSTAT0 | Set if any non-correctable errors have been encountered during the verify operation[1] |
| FERSTAT | EPVIOLIF | Set if any area of the buffer RAM EEE partition is protected |

1. As found in the memory map for FTM512K3.

### 26.4.2.9 Erase P-Flash Block Command

The Erase P-Flash Block operation will erase all addresses in a P-Flash block.

**Table 26-49. Erase P-Flash Block Command FCCOB Requirements**

| CCOBIX[2:0] | FCCOB Parameters | |
|---|---|---|
| 000 | 0x09 | Global address [22:16] to identify P-Flash block |
| 001 | Global address [15:0] in P-Flash block to be erased | |

Upon clearing CCIF to launch the Erase P-Flash Block command, the Memory Controller will erase the selected P-Flash block and verify that it is erased. The CCIF flag will set after the Erase P-Flash Block operation has completed.

**Table 26-50. Erase P-Flash Block Command Error Handling**

| Register | Error Bit | Error Condition |
|---|---|---|
| FSTAT | ACCERR | Set if CCOBIX[2:0] != 001 at command launch |
| | | Set if a Load Data Field command sequence is currently active |
| | | Set if command not available in current mode (see Table 26-30) |
| | | Set if an invalid global address [22:16] is supplied(1) |
| | FPVIOL | Set if an area of the selected P-Flash block is protected |
| | MGSTAT1 | Set if any errors have been encountered during the verify operation(2) |
| | MGSTAT0 | Set if any non-correctable errors have been encountered during the verify operation[2] |
| FERSTAT | EPVIOLIF | None |

1. As defined by the memory map for FTM512K3.
2. As found in the memory map for FTM512K3.

### 26.4.2.10 Erase P-Flash Sector Command

The Erase P-Flash Sector operation will erase all addresses in a P-Flash sector.

**Table 26-51. Erase P-Flash Sector Command FCCOB Requirements**

| CCOBIX[2:0] | FCCOB Parameters | |
|---|---|---|
| 000 | 0x0A | Global address [22:16] to identify P-Flash block to be erased |
| 001 | Global address [15:0] anywhere within the sector to be erased. Refer to Section 26.1.2.1 for the P-Flash sector size. | |

Upon clearing CCIF to launch the Erase P-Flash Sector command, the Memory Controller will erase the selected Flash sector and then verify that it is erased. The CCIF flag will be set after the Erase P-Flash Sector operation has completed.

**Table 26-52. Erase P-Flash Sector Command Error Handling**

| Register | Error Bit | Error Condition |
|---|---|---|
| FSTAT | ACCERR | Set if CCOBIX[2:0] != 001 at command launch |
| | | Set if a Load Data Field command sequence is currently active |
| | | Set if command not available in current mode (see Table 26-30) |
| | | Set if an invalid global address [22:16] is supplied(1) |
| | | Set if a misaligned phrase address is supplied (global address [2:0] != 000) |
| | FPVIOL | Set if the selected P-Flash sector is protected |
| | MGSTAT1 | Set if any errors have been encountered during the verify operation |
| | MGSTAT0 | Set if any non-correctable errors have been encountered during the verify operation |
| FERSTAT | EPVIOLIF | None |

1. As defined by the memory map for FTM512K3.

### 26.4.2.11 Unsecure Flash Command

The Unsecure Flash command will erase the entire P-Flash and D-Flash memory space and, if the erase is successful, will release security.

**Table 26-53. Unsecure Flash Command FCCOB Requirements**

| CCOBIX[2:0] | FCCOB Parameters | |
|---|---|---|
| 000 | 0x0B | Not required |

Upon clearing CCIF to launch the Unsecure Flash command, the Memory Controller will erase the entire P-Flash and D-Flash memory space and verify that it is erased. If the Memory Controller verifies that the entire Flash memory space was properly erased, security will be released. If the erase verify is not successful, the Unsecure Flash operation sets MGSTAT1 and terminates without changing the security

state. During the execution of this command (CCIF=0) the user must not write to any Flash module register. The CCIF flag is set after the Unsecure Flash operation has completed.

**Table 26-54. Unsecure Flash Command Error Handling**

| Register | Error Bit | Error Condition |
|---|---|---|
| FSTAT | ACCERR | Set if CCOBIX[2:0] != 000 at command launch |
| | | Set if a Load Data Field command sequence is currently active |
| | | Set if command not available in current mode (see Table 26-30) |
| | FPVIOL | Set if any area of the P-Flash memory is protected |
| | MGSTAT1 | Set if any errors have been encountered during the verify operation(1) |
| | MGSTAT0 | Set if any non-correctable errors have been encountered during the verify operation[1] |
| FERSTAT | EPVIOLIF | Set if any area of the buffer RAM EEE partition is protected |

1. As found in the memory map for FTM512K3.

### 26.4.2.12 Verify Backdoor Access Key Command

The Verify Backdoor Access Key command will only execute if it is enabled by the KEYEN bits in the FSEC register (see Table 26-11). The Verify Backdoor Access Key command releases security if user-supplied keys match those stored in the Flash security bytes of the Flash configuration field (see Table 26-3). The Verify Backdoor Access Key command must not be executed from the Flash block containing the backdoor comparison key to avoid code runaway.

**Table 26-55. Verify Backdoor Access Key Command FCCOB Requirements**

| CCOBIX[2:0] | FCCOB Parameters | |
|---|---|---|
| 000 | 0x0C | Not required |
| 001 | Key 0 | |
| 010 | Key 1 | |
| 011 | Key 2 | |
| 100 | Key 3 | |

Upon clearing CCIF to launch the Verify Backdoor Access Key command, the Memory Controller will check the FSEC KEYEN bits to verify that this command is enabled. If not enabled, the Memory Controller sets the ACCERR bit in the FSTAT register and terminates. If the command is enabled, the Memory Controller compares the key provided in FCCOB to the backdoor comparison key in the Flash configuration field with Key 0 compared to 0x7F_FF00, etc. If the backdoor keys match, security will be released. If the backdoor keys do not match, security is not released and all future attempts to execute the Verify Backdoor Access Key command are aborted (set ACCERR) until a reset occurs. The CCIF flag is set after the Verify Backdoor Access Key operation has completed.

**Table 26-56. Verify Backdoor Access Key Command Error Handling**

| Register | Error Bit | Error Condition |
|---|---|---|
| FSTAT | ACCERR | Set if CCOBIX[2:0] != 100 at command launch |
| | | Set if a Load Data Field command sequence is currently active |
| | | Set if an incorrect backdoor key is supplied |
| | | Set if backdoor key access has not been enabled (KEYEN[1:0] != 10, see Section 26.3.2.2) |
| | | Set if the backdoor key has mismatched since the last reset |
| | FPVIOL | None |
| | MGSTAT1 | None |
| | MGSTAT0 | None |
| FERSTAT | EPVIOLIF | None |

### 26.4.2.13 Set User Margin Level Command

The Set User Margin Level command causes the Memory Controller to set the margin level for future read operations of a specific P-Flash or D-Flash block.

**Table 26-57. Set User Margin Level Command FCCOB Requirements**

| CCOBIX[2:0] | FCCOB Parameters | |
|---|---|---|
| 000 | 0x0D | Global address [22:16] to identify the Flash block |
| 001 | Margin level setting | |

Upon clearing CCIF to launch the Set User Margin Level command, the Memory Controller will set the user margin level for the targeted block and then set the CCIF flag.

Valid margin level settings for the Set User Margin Level command are defined in Table 26-58.

**Table 26-58. Valid Set User Margin Level Settings**

| CCOB (CCOBIX=001) | Level Description |
|---|---|
| 0x0000 | Return to Normal Level |
| 0x0001 | User Margin-1 Level(1) |
| 0x0002 | User Margin-0 Level(2) |

1. Read margin to the erased state
2. Read margin to the programmed state

**Table 26-59. Set User Margin Level Command Error Handling**

| Register | Error Bit | Error Condition |
|---|---|---|
| FSTAT | ACCERR | Set if CCOBIX[2:0] != 001 at command launch |
| | | Set if a Load Data Field command sequence is currently active |
| | | Set if command not available in current mode (see Table 26-30) |
| | | Set if an invalid global address [22:16] is supplied(1) |
| | | Set if an invalid margin level setting is supplied |
| | FPVIOL | None |
| | MGSTAT1 | None |
| | MGSTAT0 | None |
| FERSTAT | EPVIOLIF | None |

1. As defined by the memory map for FTM512K3.

**NOTE**

User margin levels can be used to check that Flash memory contents have adequate margin for normal level read operations. If unexpected results are encountered when checking Flash memory contents at user margin levels, a potential loss of information has been detected.

### 26.4.2.14 Set Field Margin Level Command

The Set Field Margin Level command, valid in special modes only, causes the Memory Controller to set the margin level specified for future read operations of a specific P-Flash or D-Flash block.

**Table 26-60. Set Field Margin Level Command FCCOB Requirements**

| CCOBIX[2:0] | FCCOB Parameters | |
|---|---|---|
| 000 | 0x0E | Global address [22:16] to identify the Flash block |
| 001 | Margin level setting | |

Upon clearing CCIF to launch the Set Field Margin Level command, the Memory Controller will set the field margin level for the targeted block and then set the CCIF flag.

Valid margin level settings for the Set Field Margin Level command are defined in Table 26-61.

**Table 26-61. Valid Set Field Margin Level Settings**

| CCOB (CCOBIX=001) | Level Description |
|---|---|
| 0x0000 | Return to Normal Level |
| 0x0001 | User Margin-1 Level(1) |

**Table 26-61. Valid Set Field Margin Level Settings**

| CCOB (CCOBIX=001) | Level Description |
|---|---|
| 0x0002 | User Margin-0 Level(2) |
| 0x0003 | Field Margin-1 Level1 |
| 0x0004 | Field Margin-0 Level2 |

1. Read margin to the erased state
2. Read margin to the programmed state

**Table 26-62. Set Field Margin Level Command Error Handling**

| Register | Error Bit | Error Condition |
|---|---|---|
| FSTAT | ACCERR | Set if CCOBIX[2:0] != 001 at command launch |
| | | Set if a Load Data Field command sequence is currently active |
| | | Set if command not available in current mode (see Table 26-30) |
| | | Set if an invalid global address [22:16] is supplied(1) |
| | | Set if an invalid margin level setting is supplied |
| | FPVIOL | None |
| | MGSTAT1 | None |
| | MGSTAT0 | None |
| FERSTAT | EPVIOLIF | None |

1. As defined by the memory map for FTM512K3.

**CAUTION**

Field margin levels must only be used during verify of the initial factory programming.

**NOTE**

Field margin levels can be used to check that Flash memory contents have adequate margin for data retention at the normal level setting. If unexpected results are encountered when checking Flash memory contents at field margin levels, the Flash memory contents should be erased and reprogrammed.

### 26.4.2.15 Full Partition D-Flash Command

The Full Partition D-Flash command allows the user to allocate sectors within the D-Flash block for applications and a partition within the buffer RAM for EEPROM access. The D-Flash block consists of 128 sectors with 256 bytes per sector.

**Table 26-63. Full Partition D-Flash Command FCCOB Requirements**

| CCOBIX[2:0] | FCCOB Parameters | |
|---|---|---|
| 000 | 0x0F | Not required |
| 001 | Number of 256 byte sectors for the D-Flash user partition (DFPART) | |
| 010 | Number of 256 byte sectors for buffer RAM EEE partition (ERPART) | |

Upon clearing CCIF to launch the Full Partition D-Flash command, the following actions are taken to define a partition within the D-Flash block for direct access (DFPART) and a partition within the buffer RAM for EEE use (ERPART):

- Validate the DFPART and ERPART values provided:
  - DFPART <= 128 (maximum number of 256 byte sectors in D-Flash block)
  - ERPART <= 16 (maximum number of 256 byte sectors in buffer RAM)
  - If ERPART > 0, 128 - DFPART >= 12 (minimum number of 256 byte sectors in the D-Flash block required to support EEE)
  - If ERPART > 0, ((128-DFPART)/ERPART) >= 8 (minimum ratio of D-Flash EEE space to buffer RAM EEE space to support EEE)
- Erase the D-Flash block and the EEE nonvolatile information register
- Program DFPART to the EEE nonvolatile information register at global address 0x12_0000 (see Table 26-7)
- Program a duplicate DFPART to the EEE nonvolatile information register at global address 0x12_0002 (see Table 26-7)
- Program ERPART to the EEE nonvolatile information register at global address 0x12_0004 (see Table 26-7)
- Program a duplicate ERPART to the EEE nonvolatile information register at global address 0x12_0006 (see Table 26-7)

The D-Flash user partition will start at global address 0x10_0000. The buffer RAM EEE partition will end at global address 0x13_FFFF. After the Full Partition D-Flash operation has completed, the CCIF flag will set.

Running the Full Partition D-Flash command a second time will result in the previous partition values and the entire D-Flash memory being erased. The data value written corresponds to the number of 256 byte sectors allocated for either direct D-Flash access (DFPART) or buffer RAM EEE access (ERPART).

**Table 26-64. Full Partition D-Flash Command Error Handling**

| Register | Error Bit | Error Condition |
|---|---|---|
| FSTAT | ACCERR | Set if CCOBIX[2:0] != 010 at command launch |
| | | Set if a Load Data Field command sequence is currently active |
| | | Set if command not available in current mode (see Table 26-30) |
| | | Set if an invalid DFPART or ERPART selection is supplied |
| | FPVIOL | None |
| | MGSTAT1 | Set if any errors have been encountered during the read |
| | MGSTAT0 | Set if any non-correctable errors have been encountered during the read |
| FERSTAT | EPVIOLIF | None |

### 26.4.2.16 Erase Verify D-Flash Section Command

The Erase Verify D-Flash Section command will verify that a section of code in the D-Flash user partition is erased. The Erase Verify D-Flash Section command defines the starting point of the data to be verified and the number of words.

**Table 26-65. Erase Verify D-Flash Section Command FCCOB Requirements**

| CCOBIX[2:0] | FCCOB Parameters | |
|---|---|---|
| 000 | 0x10 | Global address [22:16] to identify the D-Flash block |
| 001 | Global address [15:0] of the first word to be verified | |
| 010 | Number of words to be verified | |

Upon clearing CCIF to launch the Erase Verify D-Flash Section command, the Memory Controller will verify the selected section of D-Flash memory is erased. The CCIF flag will set after the Erase Verify D-Flash Section operation has completed.

**Table 26-66. Erase Verify D-Flash Section Command Error Handling**

| Register | Error Bit | Error Condition |
|---|---|---|
| FSTAT | ACCERR | Set if CCOBIX[2:0] != 010 at command launch |
| | | Set if a Load Data Field command sequence is currently active |
| | | Set if command not available in current mode (see Table 26-30) |
| | | Set if an invalid global address [22:0] is supplied |
| | | Set if a misaligned word address is supplied (global address [0] != 0) |
| | | Set if the global address [22:0] points to an area of the D-Flash EEE partition |
| | | Set if the requested section breaches the end of the D-Flash block or goes into the D-Flash EEE partition |
| | FPVIOL | None |
| | MGSTAT1 | Set if any errors have been encountered during the read |
| | MGSTAT0 | Set if any non-correctable errors have been encountered during the read |
| FERSTAT | EPVIOLIF | None |

### 26.4.2.17 Program D-Flash Command

The Program D-Flash operation programs one to four previously erased words in the D-Flash user partition. The Program D-Flash operation will confirm that the targeted location(s) were successfully programmed upon completion.

**CAUTION**

A Flash word must be in the erased state before being programmed. Cumulative programming of bits within a Flash word is not allowed.

**Table 26-67. Program D-Flash Command FCCOB Requirements**

| CCOBIX[2:0] | FCCOB Parameters | |
|---|---|---|
| 000 | 0x11 | Global address [22:16] to identify the D-Flash block |
| 001 | Global address [15:0] of word to be programmed | |
| 010 | Word 0 program value | |
| 011 | Word 1 program value, if desired | |
| 100 | Word 2 program value, if desired | |
| 101 | Word 3 program value, if desired | |

Upon clearing CCIF to launch the Program D-Flash command, the user-supplied words will be transferred to the Memory Controller and be programmed. The CCOBIX index value at Program D-Flash command launch determines how many words will be programmed in the D-Flash block. No protection checks are made in the Program D-Flash operation on the D-Flash block, only access error checks. The CCIF flag is set when the operation has completed.

**Table 26-68. Program D-Flash Command Error Handling**

| Register | Error Bit | Error Condition |
|---|---|---|
| FSTAT | ACCERR | Set if CCOBIX[2:0] < 010 at command launch |
| | | Set if CCOBIX[2:0] > 101 at command launch |
| | | Set if a Load Data Field command sequence is currently active |
| | | Set if command not available in current mode (see Table 26-30) |
| | | Set if an invalid global address [22:0] is supplied |
| | | Set if a misaligned word address is supplied (global address [0] != 0) |
| | | Set if the global address [22:0] points to an area in the D-Flash EEE partition |
| | | Set if the requested group of words breaches the end of the D-Flash block or goes into the D-Flash EEE partition |
| | FPVIOL | None |
| | MGSTAT1 | Set if any errors have been encountered during the verify operation |
| | MGSTAT0 | Set if any non-correctable errors have been encountered during the verify operation |
| FERSTAT | EPVIOLIF | None |

### 26.4.2.18 Erase D-Flash Sector Command

The Erase D-Flash Sector operation will erase all addresses in a sector of the D-Flash user partition.

**Table 26-69. Erase D-Flash Sector Command FCCOB Requirements**

| CCOBIX[2:0] | FCCOB Parameters | |
|---|---|---|
| 000 | 0x12 | Global address [22:16] to identify D-Flash block |
| 001 | Global address [15:0] anywhere within the sector to be erased. See Section 26.1.2.2 for D-Flash sector size. | |

Upon clearing CCIF to launch the Erase D-Flash Sector command, the Memory Controller will erase the selected Flash sector and verify that it is erased. The CCIF flag will set after the Erase D-Flash Sector operation has completed.

**Table 26-70. Erase D-Flash Sector Command Error Handling**

| Register | Error Bit | Error Condition |
|---|---|---|
| FSTAT | ACCERR | Set if CCOBIX[2:0] != 001 at command launch |
| | | Set if a Load Data Field command sequence is currently active |
| | | Set if command not available in current mode (see Table 26-30) |
| | | Set if an invalid global address [22:0] is supplied |
| | | Set if a misaligned word address is supplied (global address [0] != 0) |
| | | Set if the global address [22:0] points to the D-Flash EEE partition |
| | FPVIOL | None |
| | MGSTAT1 | Set if any errors have been encountered during the verify operation |
| | MGSTAT0 | Set if any non-correctable errors have been encountered during the verify operation |
| FERSTAT | EPVIOLIF | None |

### 26.4.2.19 Enable EEPROM Emulation Command

The Enable EEPROM Emulation command causes the Memory Controller to enable EEE activity. EEE activity is disabled after any reset.

**Table 26-71. Enable EEPROM Emulation Command FCCOB Requirements**

| CCOBIX[2:0] | FCCOB Parameters | |
|---|---|---|
| 000 | 0x13 | Not required |

Upon clearing CCIF to launch the Enable EEPROM Emulation command, the CCIF flag will set after the Memory Controller enables EEE operations using the contents of the EEE tag RAM and tag counter. The Full Partition D-Flash or the Partition D-Flash command must be run prior to launching the Enable EEPROM Emulation command.

**Table 26-72. Enable EEPROM Emulation Command Error Handling**

| Register | Error Bit | Error Condition |
|---|---|---|
| FSTAT | ACCERR | Set if CCOBIX[2:0] != 000 at command launch |
| | | Set if a Load Data Field command sequence is currently active |
| | | Set if Full Partition D-Flash or Partition D-Flash command not previously run |
| | FPVIOL | None |
| | MGSTAT1 | None |
| | MGSTAT0 | None |
| FERSTAT | EPVIOLIF | None |

### 26.4.2.20 Disable EEPROM Emulation Command

The Disable EEPROM Emulation command causes the Memory Controller to suspend current EEE activity.

**Table 26-73. Disable EEPROM Emulation Command FCCOB Requirements**

| CCOBIX[2:0] | FCCOB Parameters | |
|---|---|---|
| 000 | 0x14 | Not required |

Upon clearing CCIF to launch the Disable EEPROM Emulation command, the Memory Controller will halt EEE operations at the next convenient point without clearing the EEE tag RAM or tag counter before setting the CCIF flag.

**Table 26-74. Disable EEPROM Emulation Command Error Handling**

| Register | Error Bit | Error Condition |
|---|---|---|
| FSTAT | ACCERR | Set if CCOBIX[2:0] != 000 at command launch |
| | | Set if a Load Data Field command sequence is currently active |
| | | Set if Full Partition D-Flash or Partition D-Flash command not previously run |
| | FPVIOL | None |
| | MGSTAT1 | None |
| | MGSTAT0 | None |
| FERSTAT | EPVIOLIF | None |

### 26.4.2.21 EEPROM Emulation Query Command

The EEPROM Emulation Query command returns EEE partition and status variables.

**Table 26-75. EEPROM Emulation Query Command FCCOB Requirements**

| CCOBIX[2:0] | FCCOB Parameters | |
|---|---|---|
| 000 | 0x15 | Not required |
| 001 | Return DFPART | |
| 010 | Return ERPART | |
| 011 | Return ECOUNT(1) | |
| 100 | Return Dead Sector Count | Return Ready Sector Count |

1. Indicates sector erase count

Upon clearing CCIF to launch the EEPROM Emulation Query command, the CCIF flag will set after the EEE partition and status variables are stored in the FCCOBIX register.If the Emulation Query command is executed prior to partitioning (Partition D-Flash Command Section 26.4.2.15), the following reset values are returned: DFPART = 0x_FFFF, ERPART = 0x_FFFF, ECOUNT = 0x_FFFF, Dead Sector Count = 0x_00, Ready Sector Count = 0x_00.

**Table 26-76. EEPROM Emulation Query Command Error Handling**

| Register | Error Bit | Error Condition |
|---|---|---|
| FSTAT | ACCERR | Set if CCOBIX[2:0] != 000 at command launch |
| | | Set if a Load Data Field command sequence is currently active |
| | | Set if command not available in current mode (see Table 26-30) |
| | FPVIOL | None |
| | MGSTAT1 | None |
| | MGSTAT0 | None |
| FERSTAT | EPVIOLIF | None |

### 26.4.2.22 Partition D-Flash Command

The Partition D-Flash command allows the user to allocate sectors within the D-Flash block for applications and a partition within the buffer RAM for EEPROM access. The D-Flash block consists of 128 sectors with 256 bytes per sector. The Erase All Blocks command must be run prior to launching the Partition D-Flash command.

**Table 26-77. Partition D-Flash Command FCCOB Requirements**

| CCOBIX[2:0] | FCCOB Parameters | |
|---|---|---|
| 000 | 0x20 | Not required |
| 001 | Number of 256 byte sectors for the D-Flash user partition (DFPART) | |
| 010 | Number of 256 byte sectors for buffer RAM EEE partition (ERPART) | |

Upon clearing CCIF to launch the Partition D-Flash command, the following actions are taken to define a partition within the D-Flash block for direct access (DFPART) and a partition within the buffer RAM for EEE use (ERPART):

- Validate the DFPART and ERPART values provided:
  - DFPART <= 128 (maximum number of 256 byte sectors in D-Flash block)
  - ERPART <= 16 (maximum number of 256 byte sectors in buffer RAM)
  - If ERPART > 0, 128 - DFPART >= 12 (minimum number of 256 byte sectors in the D-Flash block required to support EEE)
  - If ERPART > 0, ((128-DFPART)/ERPART) >= 8 (minimum ratio of D-Flash EEE space to buffer RAM EEE space to support EEE)
- Erase verify the D-Flash block and the EEE nonvolatile information register
- Program DFPART to the EEE nonvolatile information register at global address 0x12_0000 (see Table 26-7)

- Program a duplicate DFPART to the EEE nonvolatile information register at global address 0x12_0002 (see Table 26-7)
- Program ERPART to the EEE nonvolatile information register at global address 0x12_0004 (see Table 26-7)
- Program a duplicate ERPART to the EEE nonvolatile information register at global address 0x12_0006 (see Table 26-7)

The D-Flash user partition will start at global address 0x10_0000. The buffer RAM EEE partition will end at global address 0x13_FFFF. After the Partition D-Flash operation has completed, the CCIF flag will set.

Running the Partition D-Flash command a second time will result in the ACCERR bit within the FSTAT register being set. The data value written corresponds to the number of 256 byte sectors allocated for either direct D-Flash access (DFPART) or buffer RAM EEE access (ERPART).

**Table 26-78. Partition D-Flash Command Error Handling**

| Register | Error Bit | Error Condition |
|---|---|---|
| FSTAT | ACCERR | Set if CCOBIX[2:0] != 010 at command launch |
| | | Set if a Load Data Field command sequence is currently active |
| | | Set if command not available in current mode (see Table 26-30) |
| | | Set if partitions have already been defined |
| | | Set if an invalid DFPART or ERPART selection is supplied |
| | FPVIOL | None |
| | MGSTAT1 | Set if any errors have been encountered during the read |
| | MGSTAT0 | Set if any non-correctable errors have been encountered during the read |
| FERSTAT | EPVIOLIF | None |

## 26.4.3 Interrupts

The Flash module can generate an interrupt when a Flash command operation has completed or when a Flash command operation has detected an EEE error or an ECC fault.

**Table 26-79. Flash Interrupt Sources**

| Interrupt Source | Interrupt Flag | Local Enable | Global (CCR) Mask |
|---|---|---|---|
| Flash Command Complete | CCIF (FSTAT register) | CCIE (FCNFG register) | I Bit |
| Flash EEE Erase Error | ERSERIF (FERSTAT register) | ERSERIE (FERCNFG register) | I Bit |
| Flash EEE Program Error | PGMERIF (FERSTAT register) | PGMERIE (FERCNFG register) | I Bit |
| Flash EEE Protection Violation | EPVIOLIF (FERSTAT register) | EPVIOLIE (FERCNFG register) | I Bit |
| Flash EEE Error Type 1 Violation | ERSVIF1 (FERSTAT register) | ERSVIE1 (FERCNFG register) | I Bit |
| Flash EEE Error Type 0 Violation | ERSVIF0 (FERSTAT register) | ERSVIE0 (FERCNFG register) | I Bit |
| ECC Double Bit Fault on Flash Read | DFDIF (FERSTAT register) | DFDIE (FERCNFG register) | I Bit |
| ECC Single Bit Fault on Flash Read | SFDIF (FERSTAT register) | SFDIE (FERCNFG register) | I Bit |

**NOTE**

Vector addresses and their relative interrupt priority are determined at the MCU level.

### 26.4.3.1 Description of Flash Interrupt Operation

The Flash module uses the CCIF flag in combination with the CCIE interrupt enable bit to generate the Flash command interrupt request. The Flash module uses the ERSEIF, PGMEIF, EPVIOLIF, ERSVIF1, ERSVIF0, DFDIF and SFDIF flags in combination with the ERSEIE, PGMEIE, EPVIOLIE, ERSVIE1, ERSVIE0, DFDIE and SFDIE interrupt enable bits to generate the Flash error interrupt request. For a detailed description of the register bits involved, refer to Section 26.3.2.5, “Flash Configuration Register (FCNFG)”, Section 26.3.2.6, “Flash Error Configuration Register (FERCNFG)”, Section 26.3.2.7, “Flash Status Register (FSTAT)”, and Section 26.3.2.8, “Flash Error Status Register (FERSTAT)”.

The logic used for generating the Flash module interrupts is shown in Figure 26-27.

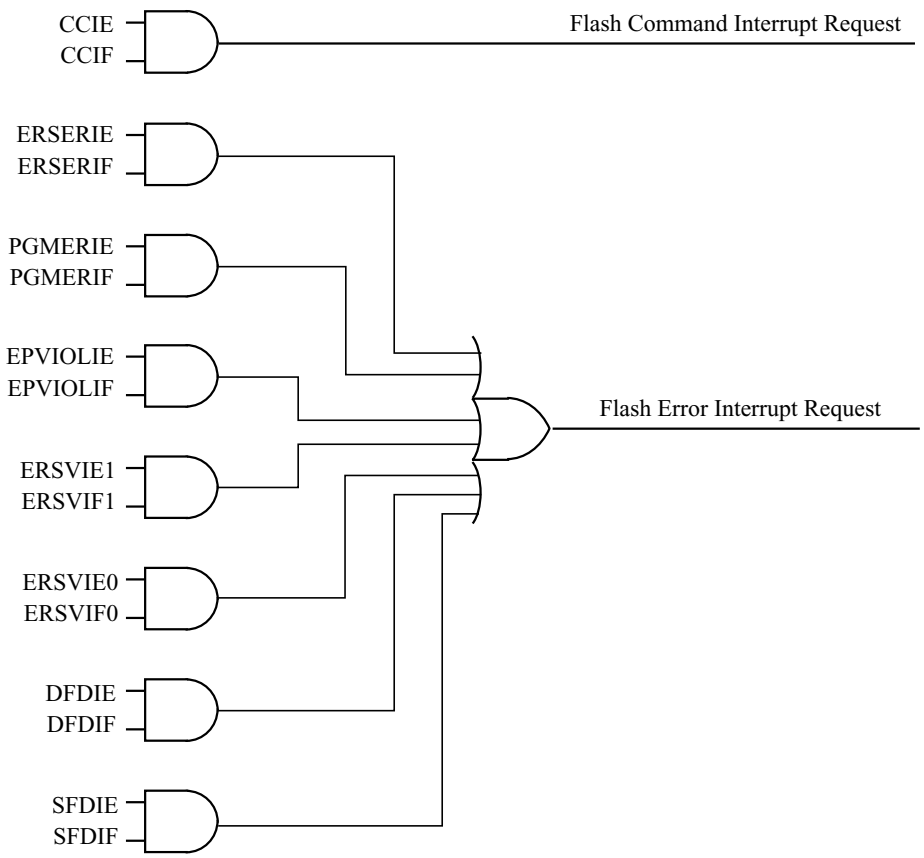



**Figure 26-27. Flash Module Interrupts Implementation**

### 26.4.4 Wait Mode

The Flash module is not affected if the MCU enters wait mode. The Flash module can recover the MCU from wait via the CCIF interrupt (see Section 26.4.3, “Interrupts”).

### 26.4.5 Stop Mode

If a Flash command is active (CCIF = 0) or an EE-Emulation operation is pending when the MCU requests stop mode, the current Flash operation will be completed before the CPU is allowed to enter stop mode.

## 26.5 Security

The Flash module provides security information to the MCU. The Flash security state is defined by the SEC bits of the FSEC register (see Table 26-12). During reset, the Flash module initializes the FSEC register using data read from the security byte of the Flash configuration field at global address 0x7F_FF0F.

The security state out of reset can be permanently changed by programming the security byte of the Flash configuration field. This assumes that you are starting from a mode where the necessary P-Flash erase and program commands are available and that the upper region of the P-Flash is unprotected. If the Flash security byte is successfully programmed, its new value will take affect after the next MCU reset.

The following subsections describe these security-related subjects:

- Unsecuring the MCU using Backdoor Key Access
- Unsecuring the MCU in Special Single Chip Mode using BDM
- Mode and Security Effects on Flash Command Availability

### 26.5.1 Unsecuring the MCU using Backdoor Key Access

The MCU may be unsecured by using the backdoor key access feature which requires knowledge of the contents of the backdoor keys (four 16-bit words programmed at addresses 0x7F_FF00–0x7F_FF07). If the KEYEN[1:0] bits are in the enabled state (see Section 26.3.2.2), the Verify Backdoor Access Key command (see Section 26.4.2.12) allows the user to present four prospective keys for comparison to the keys stored in the Flash memory via the Memory Controller. If the keys presented in the Verify Backdoor Access Key command match the backdoor keys stored in the Flash memory, the SEC bits in the FSEC register (see Table 26-12) will be changed to unsecure the MCU. Key values of 0x0000 and 0xFFFF are not permitted as backdoor keys. While the Verify Backdoor Access Key command is active, P-Flash block 0 will not be available for read access and will return invalid data.

The user code stored in the P-Flash memory must have a method of receiving the backdoor keys from an external stimulus. This external stimulus would typically be through one of the on-chip serial ports.

If the KEYEN[1:0] bits are in the enabled state (see Section 26.3.2.2), the MCU can be unsecured by the backdoor key access sequence described below:

1. Follow the command sequence for the Verify Backdoor Access Key command as explained in Section 26.4.2.12
2. If the Verify Backdoor Access Key command is successful, the MCU is unsecured and the SEC[1:0] bits in the FSEC register are forced to the unsecure state of 10

The Verify Backdoor Access Key command is monitored by the Memory Controller and an illegal key will prohibit future use of the Verify Backdoor Access Key command. A reset of the MCU is the only method to re-enable the Verify Backdoor Access Key command.

After the backdoor keys have been correctly matched, the MCU will be unsecured. After the MCU is unsecured, the sector containing the Flash security byte can be erased and the Flash security byte can be reprogrammed to the unsecure state, if desired.

In the unsecure state, the user has full control of the contents of the backdoor keys by programming addresses 0x7F_FF00–0x7F_FF07 in the Flash configuration field.

The security as defined in the Flash security byte (0x7F_FF0F) is not changed by using the Verify Backdoor Access Key command sequence. The backdoor keys stored in addresses 0x7F_FF00–0x7F_FF07 are unaffected by the Verify Backdoor Access Key command sequence. After the next reset of the MCU, the security state of the Flash module is determined by the Flash security byte

(0x7F_FF0F). The Verify Backdoor Access Key command sequence has no effect on the program and erase protections defined in the Flash protection register, FPROT.

### 26.5.2 Unsecuring the MCU in Special Single Chip Mode using BDM

The MCU can be unsecured in special single chip mode by erasing the P-Flash and D-Flash memory by one of the following methods:

- Reset the MCU into special single chip mode, delay while the erase test is performed by the BDM, send BDM commands to disable protection in the P-Flash and D-Flash memory, and execute the Erase All Blocks command write sequence to erase the P-Flash and D-Flash memory.
- Reset the MCU into special expanded wide mode, disable protection in the P-Flash and D-Flash memory and run code from external memory to execute the Erase All Blocks command write sequence to erase the P-Flash and D-Flash memory.

After the CCIF flag sets to indicate that the Erase All Blocks operation has completed, reset the MCU into special single chip mode. The BDM will execute the Erase Verify All Blocks command write sequence to verify that the P-Flash and D-Flash memory is erased. If the P-Flash and D-Flash memory are verified as erased the MCU will be unsecured. All BDM commands will be enabled and the Flash security byte may be programmed to the unsecure state by the following method:

- Send BDM commands to execute a ‘Program P-Flash’ command sequence to program the Flash security byte to the unsecured state and reset the MCU.

### 26.5.3 Mode and Security Effects on Flash Command Availability

The availability of Flash module commands depends on the MCU operating mode and security state as shown in Table 26-30.

## 26.6 Initialization

On each system reset the Flash module executes a reset sequence which establishes initial values for the Flash Block Configuration Parameters, the FPROT and DFPROT protection registers, and the FOPT and FSEC registers. The Flash module reverts to built-in default values that leave the module in a fully protected and secured state if errors are encountered during execution of the reset sequence. If a double bit fault is detected during the reset sequence, both MGSTAT bits in the FSTAT register will be set. The ACCERR bit in the FSTAT register is set if errors are encountered while initializing the EEE buffer ram during the reset sequence.

CCIF remains clear throughout the reset sequence. The Flash module holds off all CPU access for the initial portion of the reset sequence. While Flash reads are possible when the hold is removed, writes to the FCCOBIX, FCCOBHI, and FCCOBLO registers are ignored to prevent command activity while the Memory Controller remains busy. Completion of the reset sequence is marked by setting CCIF high which enables writes to the FCCOBIX, FCCOBHI, and FCCOBLO registers to launch any available Flash command.

If a reset occurs while any Flash command is in progress, that command will be immediately aborted. The state of the word being programmed or the sector/block being erased is not guaranteed.

# Chapter 27
# 512 KByte Flash Module (S12XFTM512K3V1)

**Table 27-1. Revision History**

| Revision Number | Revision Date | Sections Affected | Description of Changes |
|---|---|---|---|
| V01.09 | 14 Nov 2007 | 27.5.2/27-1075<br><br>27.4.2/27-1051<br><br><br>27.4.2.8/27-1057 | - Changed terminology from ‘word program’ to “Program P-Flash’ in the BDM unsecuring description, Section 27.5.2<br>- Added requirement that user not write any Flash module register during execution of commands ‘Erase All Blocks’, Section 27.4.2.8, and ‘Unsecure Flash’, Section 27.4.2.11<br>- Added statement that security is released upon successful completion of command ‘Erase All Blocks’, Section 27.4.2.8 |
| V01.10 | 19 Dec 2007 | 27.4.2/27-1051 | - Corrected Error Handling table for Full Partition D-Flash, Partition D-Flash, and EEPROM Emulation Query commands |
| V01.11 | 25 Sep 2009 | 27.1/27-1016<br>27.3.2.1/27-1027<br>27.4.2.4/27-1054<br><br>27.4.2.7/27-1056<br><br>27.4.2.12/27-1060<br><br>27.4.2.12/27-1060<br>27.4.2.12/27-1060<br>27.4.2.20/27-1069<br><br><br>27.3.2/27-1025<br>27.3.2.1/27-1027<br>27.4.1.2/27-1046<br>27.6/27-1075 | - Clarify single bit fault correction for P-Flash phrase<br>- Expand FDIV vs OSCCLK Frequency table<br>- Add statement concerning code runaway when executing Read Once command from Flash block containing associated fields<br>- Add statement concerning code runaway when executing Program Once command from Flash block containing associated fields<br>- Add statement concerning code runaway when executing Verify Backdoor Access Key command from Flash block containing associated fields<br>- Relate Key 0 to associated Backdoor Comparison Key address<br>- Change “power down reset” to “reset”<br>- Add ACCERR condition for Disable EEPROM Emulation command<br>The following changes were made to clarify module behavior related to Flash register access during reset sequence and while Flash commands are active:<br>- Add caution concerning register writes while command is active<br>- Writes to FCLKDIV are allowed during reset sequence while CCIF is clear<br>- Add caution concerning register writes while command is active<br>- Writes to FCCOBIX, FCCOBHI, FCCOBLO registers are ignored during reset sequence |

## 27.1 Introduction

The FTM512K3 module implements the following:

- 512 Kbytes of P-Flash (Program Flash) memory, consisting of 3 physical Flash blocks, intended primarily for nonvolatile code storage
- 32 Kbytes of D-Flash (Data Flash) memory, consisting of 1 physical Flash block, that can be used as nonvolatile storage to support the built-in hardware scheme for emulated EEPROM, as basic Flash memory primarily intended for nonvolatile data storage, or as a combination of both
- 4 Kbytes of buffer RAM, consisting of 1 physical RAM block, that can be used as emulated EEPROM using a built-in hardware scheme, as basic RAM, or as a combination of both

The Flash memory is ideal for single-supply applications allowing for field reprogramming without requiring external high voltage sources for program or erase operations. The Flash module includes a memory controller that executes commands to modify Flash memory contents or configure module resources for emulated EEPROM operation. The user interface to the memory controller consists of the indexed Flash Common Command Object (FCCOB) register which is written to with the command, global address, data, and any required command parameters. The memory controller must complete the execution of a command before the FCCOB register can be written to with a new command.

**CAUTION**

A Flash word or phrase must be in the erased state before being programmed. Cumulative programming of bits within a Flash word or phrase is not allowed.

The RAM and Flash memory may be read as bytes, aligned words, or misaligned words. Read access time is one bus cycle for bytes and aligned words, and two bus cycles for misaligned words. For Flash memory, an erased bit reads 1 and a programmed bit reads 0.

It is not possible to read from a Flash block while any command is executing on that specific Flash block. It is possible to read from a Flash block while a command is executing on a different Flash block.

Both P-Flash and D-Flash memories are implemented with Error Correction Codes (ECC) that can resolve single bit faults and detect double bit faults. For P-Flash memory, the ECC implementation requires that programming be done on an aligned 8 byte basis (a Flash phrase). Since P-Flash memory is always read by phrase, only one single bit fault in the phrase containing the byte or word accessed will be corrected.

### 27.1.1 Glossary

**Buffer RAM** — The buffer RAM constitutes the volatile memory store required for EEE. Memory space in the buffer RAM not required for EEE can be partitioned to provide volatile memory space for applications.

**Command Write Sequence** — An MCU instruction sequence to execute built-in algorithms (including program and erase) on the Flash memory.

**D-Flash Memory** — The D-Flash memory constitutes the nonvolatile memory store required for EEE. Memory space in the D-Flash memory not required for EEE can be partitioned to provide nonvolatile memory space for applications.

**D-Flash Sector** — The D-Flash sector is the smallest portion of the D-Flash memory that can be erased. The D-Flash sector consists of four 64 byte rows for a total of 256 bytes.

**EEE (Emulated EEPROM)** — A method to emulate the small sector size features and endurance characteristics associated with an EEPROM.

**EEE IFR** — Nonvolatile information register located in the D-Flash block that contains data required to partition the D-Flash memory and buffer RAM for EEE. The EEE IFR is visible in the global memory map by setting the EEEIFRON bit in the MMCCTL1 register.

**NVM Command Mode** — An NVM mode using the CPU to setup the FCCOB register to pass parameters required for Flash command execution.

**Phrase** — An aligned group of four 16-bit words within the P-Flash memory. Each phrase includes eight ECC bits for single bit fault correction and double bit fault detection within the phrase.

**P-Flash Memory** — The P-Flash memory constitutes the main nonvolatile memory store for applications.

**P-Flash Sector** — The P-Flash sector is the smallest portion of the P-Flash memory that can be erased. Each P-Flash sector contains 1024 bytes.

**Program IFR** — Nonvolatile information register located in the P-Flash block that contains the Device ID, Version ID, and the Program Once field. The Program IFR is visible in the global memory map by setting the PGMIFRON bit in the MMCCTL1 register.

## 27.1.2 Features

### 27.1.2.1 P-Flash Features

- 512 Kbytes of P-Flash memory composed of one 256 Kbyte Flash block and two 128 Kbyte Flash blocks. The 256 Kbyte Flash block consists of two 128 Kbyte sections each divided into 128 sectors of 1024 bytes. The 128 Kbyte Flash blocks are each divided into 128 sectors of 1024 bytes.
- Single bit fault correction and double bit fault detection within a 64-bit phrase during read operations
- Automated program and erase algorithm with verify and generation of ECC parity bits
- Fast sector erase and phrase program operation
- Ability to program up to one phrase in each P-Flash block simultaneously
- Flexible protection scheme to prevent accidental program or erase of P-Flash memory

#### 27.1.2.2 D-Flash Features

- Up to 32 Kbytes of D-Flash memory with 256 byte sectors for user access
- Dedicated commands to control access to the D-Flash memory over EEE operation
- Single bit fault correction and double bit fault detection within a word during read operations
- Automated program and erase algorithm with verify and generation of ECC parity bits
- Fast sector erase and word program operation
- Ability to program up to four words in a burst sequence

#### 27.1.2.3 Emulated EEPROM Features

- Up to 4 Kbytes of emulated EEPROM (EEE) accessible as 4 Kbytes of RAM
- Flexible protection scheme to prevent accidental program or erase of data
- Automatic EEE file handling using an internal Memory Controller
- Automatic transfer of valid EEE data from D-Flash memory to buffer RAM on reset
- Ability to monitor the number of outstanding EEE related buffer RAM words left to be programmed into D-Flash memory
- Ability to disable EEE operation and allow priority access to the D-Flash memory
- Ability to cancel all pending EEE operations and allow priority access to the D-Flash memory

#### 27.1.2.4 User Buffer RAM Features

- Up to 4 Kbytes of RAM for user access

#### 27.1.2.5 Other Flash Module Features

- No external high-voltage power supply required for Flash memory program and erase operations
- Interrupt generation on Flash command completion and Flash error detection
- Security mechanism to prevent unauthorized access to the Flash memory

### 27.1.3 Block Diagram

The block diagram of the Flash module is shown in Figure 27-1.

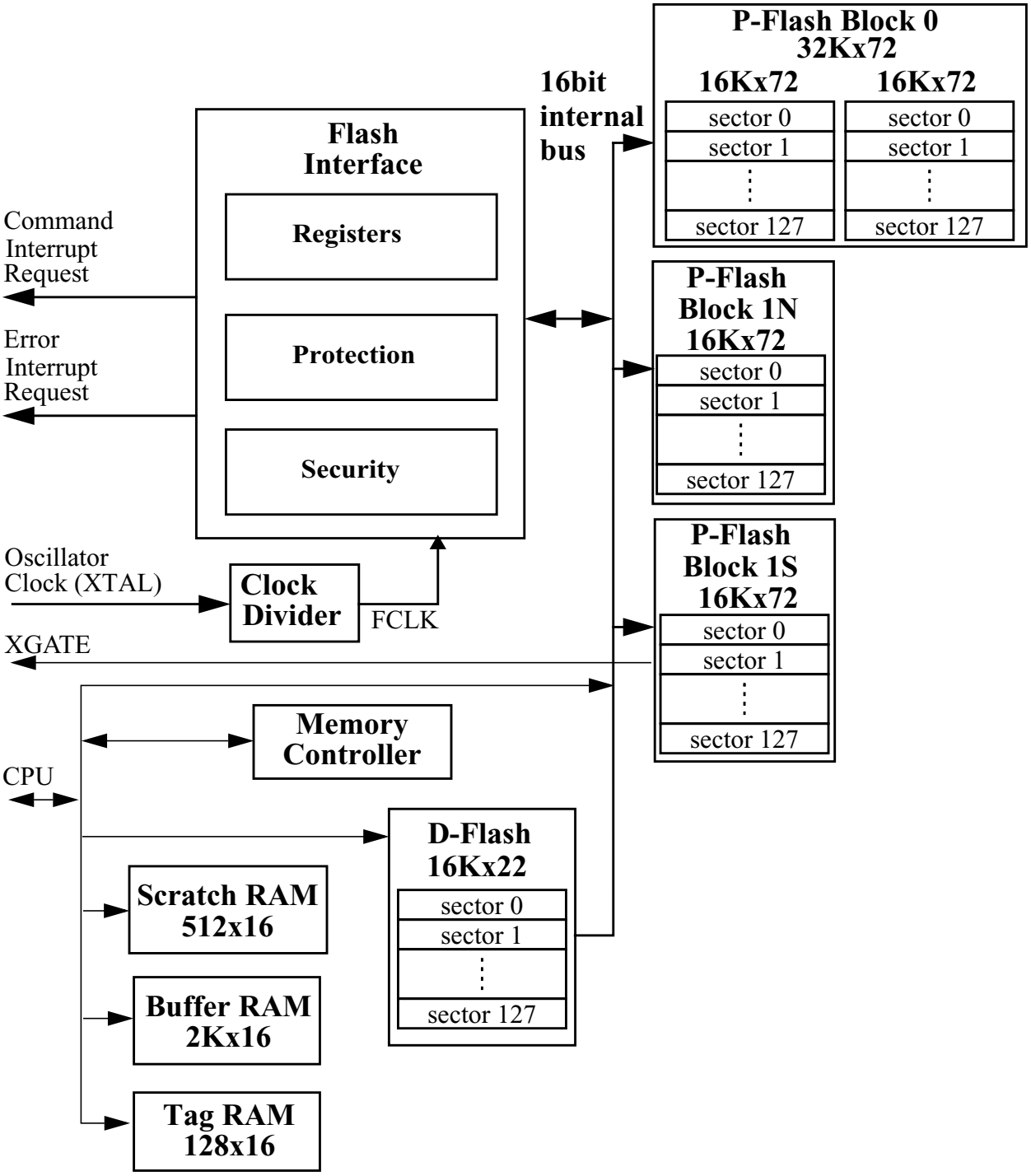



**Figure 27-1. FTM512K3 Block Diagram**

## 27.2 External Signal Description

The Flash module contains no signals that connect off-chip.

## 27.3 Memory Map and Registers

This section describes the memory map and registers for the Flash module. Read data from unimplemented memory space in the Flash module is undefined. Write access to unimplemented or reserved memory space in the Flash module will be ignored by the Flash module.

### 27.3.1 Module Memory Map

The S12X architecture places the P-Flash memory between global addresses 0x78_0000 and 0x7F_FFFF as shown in Table 27-2. The P-Flash memory map is shown in Figure 27-2.

**Table 27-2. P-Flash Memory Addressing**

| Global Address | Size (Bytes) | Description |
|---|---|---|
| 0x7C_0000 – 0x7F_FFFF | 256 K | P-Flash Block 0<br>Contains Flash Configuration Field (see Table 27-3) |
| 0x7A_0000 – 0x7B_FFFF | 128 K | P-Flash Block 1N |
| 0x78_0000 – 0x79_FFFF | 128 K | P-Flash Block 1S |

The FPROT register, described in Section 27.3.2.9, can be set to protect regions in the Flash memory from accidental program or erase. Three separate memory regions, one growing upward from global address 0x7F_8000 in the Flash memory (called the lower region), one growing downward from global address 0x7F_FFFF in the Flash memory (called the higher region), and the remaining addresses in the Flash memory, can be activated for protection. The Flash memory addresses covered by these protectable regions are shown in the P-Flash memory map. The higher address region is mainly targeted to hold the boot loader code since it covers the vector space. Default protection settings as well as security information that allows the MCU to restrict access to the Flash module are stored in the Flash configuration field as described in Table 27-3.

**Table 27-3. Flash Configuration Field(1)**

| Global Address | Size (Bytes) | Description |
|---|---|---|
| 0x7F_FF00 – 0x7F_FF07 | 8 | Backdoor Comparison Key<br>Refer to Section 27.4.2.12, “Verify Backdoor Access Key Command,” and Section 27.5.1, “Unsecuring the MCU using Backdoor Key Access” |
| 0x7F_FF08 – 0x7F_FF0B(2) | 4 | Reserved |
| 0x7F_FF0C[2] | 1 | P-Flash Protection byte.<br>Refer to Section 27.3.2.9, “P-Flash Protection Register (FPROT)” |
| 0x7F_FF0D[2] | 1 | EEE Protection byte<br>Refer to Section 27.3.2.10, “EEE Protection Register (EPROT)” |
| 0x7F_FF0E[2] | 1 | Flash Nonvolatile byte<br>Refer to Section 27.3.2.14, “Flash Option Register (FOPT)” |

**Table 27-3. Flash Configuration Field[1]**

| Global Address | Size (Bytes) | Description |
|---|---|---|
| 0x7F_FF0F[2] | 1 | Flash Security byte<br>Refer to Section 27.3.2.2, “Flash Security Register (FSEC)” |

1. Older versions may have swapped protection byte addresses
2. 0x7FF08 - 0x7F_FF0F form a Flash phrase and must be programmed in a single command write sequence. Each byte in the 0x7F_FF08 - 0x7F_FF0B reserved field should be programmed to 0xFF.

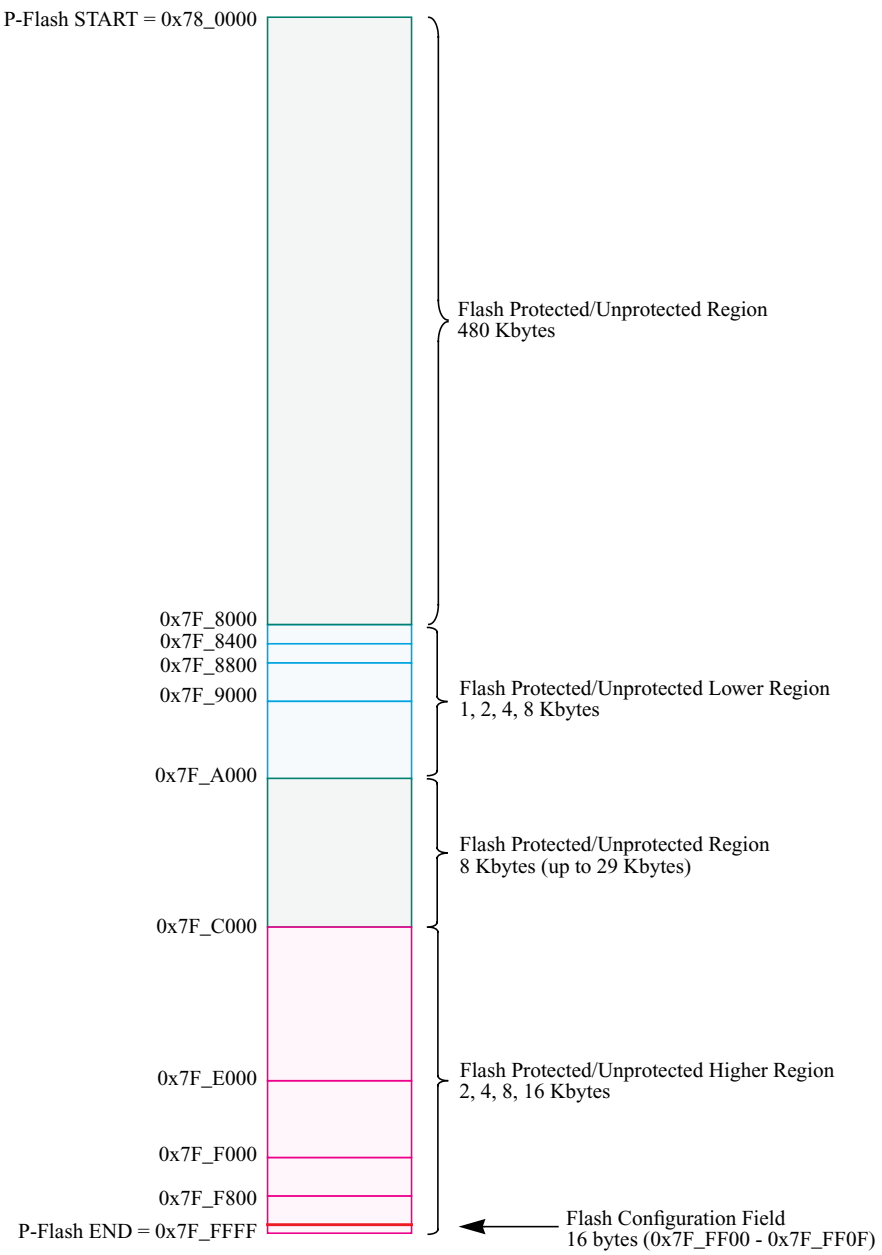



**Figure 27-2. P-Flash Memory Map**

**Table 27-4. Program IFR Fields**

| Global Address (PGMIFRON) | Size (Bytes) | Field Description |
|---|---|---|
| 0x40_0000 – 0x40_0007 | 8 | Device ID |
| 0x40_0008 – 0x40_00E7 | 224 | Reserved |
| 0x40_00E8 – 0x40_00E9 | 2 | Version ID |
| 0x40_00EA – 0x40_00FF | 22 | Reserved |
| 0x40_0100 – 0x40_013F | 64 | Program Once Field<br>Refer to Section 27.4.2.7, “Program Once Command” |
| 0x40_0140 – 0x40_01FF | 192 | Reserved |

**Table 27-5. P-Flash IFR Accessibility**

| Global Address (PGMIFRON) | Size (Bytes) | Accessed From |
|---|---|---|
| 0x40_0000 – 0x40_01FF | 512 | XBUS0 (PBLK0S)(1) |
| 0x40_0200 – 0x40_03FF | 512 | Unimplemented |
| 0x40_0400 – 0x40_05FF | 512 | XBUS0 (PBLK1N) |
| 0x40_0600 – 0x40_07FF | 512 | XBUS1 (PBLK1S) |

1. Refer to Table 27-4 for more details.

**Table 27-6. EEE Resource Fields**

| Global Address | Size (Bytes) | Description |
|---|---|---|
| 0x10_0000 – 0x10_7FFF | 32,768 | D-Flash Memory (User and EEE) |
| 0x10_8000 – 0x11_FFFF | 98,304 | Reserved |
| 0x12_0000 – 0x12_007F | 128 | EEE Nonvolatile Information Register (EEEIFRON(1) = 1) |
| 0x12_0080 – 0x12_0FFF | 3,968 | Reserved |
| 0x12_1000 – 0x12_1EFF | 3,840 | Reserved |
| 0x12_1F00 – 0x12_1FFF | 256 | EEE Tag RAM (TMGRAMON[1] = 1) |
| 0x12_2000 – 0x12_3BFF | 7,168 | Reserved |
| 0x12_3C00 – 0x12_3FFF | 1,024 | Memory Controller Scratch RAM (TMGRAMON[1] = 1) |
| 0x12_4000 – 0x12_DFFF | 40,960 | Reserved |
| 0x12_E000 – 0x12_FFFF | 8,192 | Reserved |
| 0x13_0000 – 0x13_EFFF | 61,440 | Reserved |
| 0x13_F000 – 0x13_FFFF | 4,096 | Buffer RAM (User and EEE) |

1. MMCCTL1 register bit

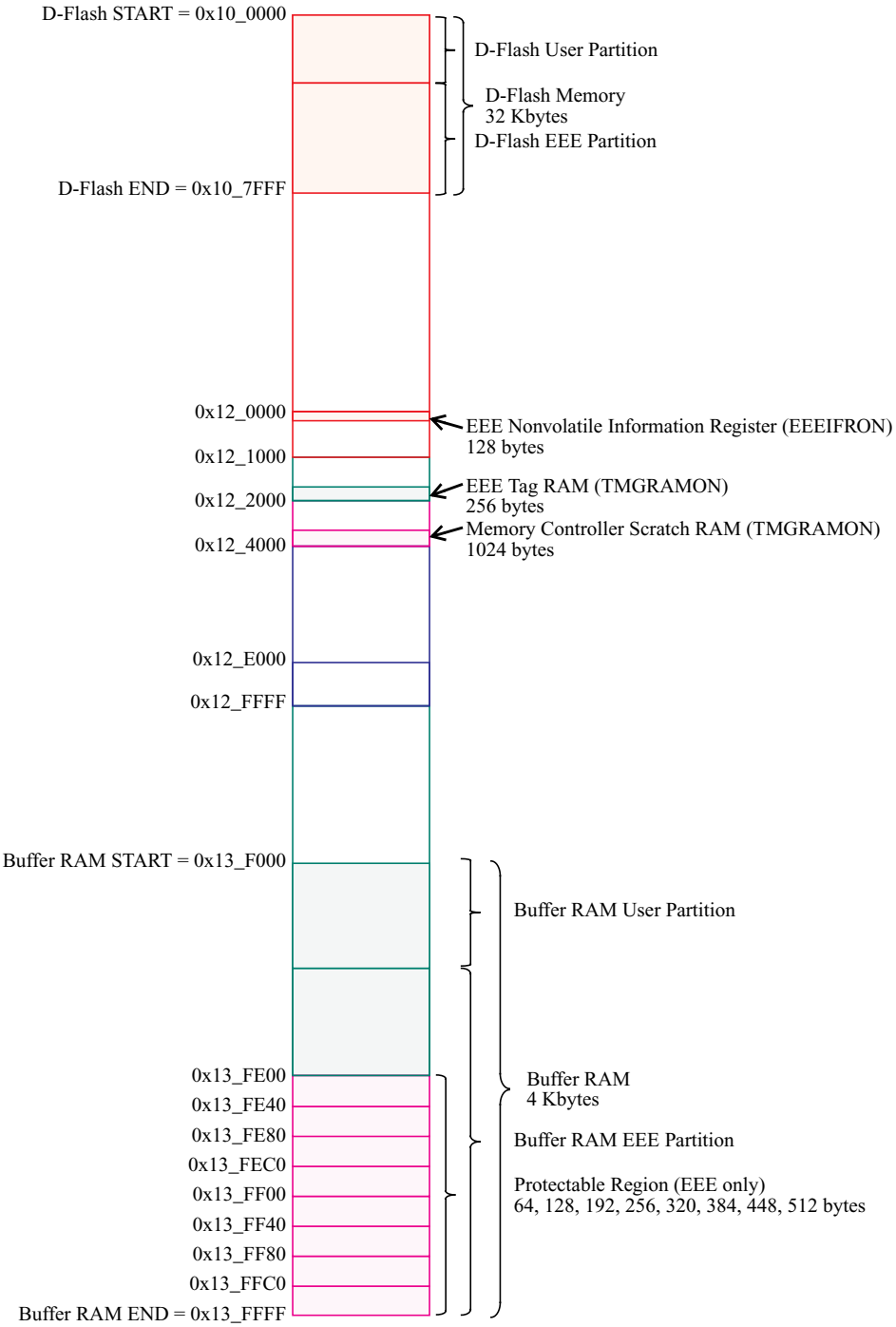



**Figure 27-3. EEE Resource Memory Map**

The Full Partition D-Flash command (see Section 27.4.2.15) is used to program the EEE nonvolatile information register fields where address 0x12_0000 defines the D-Flash partition for user access and address 0x12_0004 defines the buffer RAM partition for EEE operations.

**Table 27-7. EEE Nonvolatile Information Register Fields**

| Global Address (EEEIFRON) | Size (Bytes) | Description |
|---|---|---|
| 0x12_0000 – 0x12_0001 | 2 | D-Flash User Partition (DFPART)<br>Refer to Section 27.4.2.15, “Full Partition D-Flash Command” |
| 0x12_0002 – 0x12_0003 | 2 | D-Flash User Partition (duplicate[(1)]) |
| 0x12_0004 – 0x12_0005 | 2 | Buffer RAM EEE Partition (ERPART)<br>Refer to Section 27.4.2.15, “Full Partition D-Flash Command” |
| 0x12_0006 – 0x12_0007 | 2 | Buffer RAM EEE Partition (duplicate[1]) |
| 0x12_0008 – 0x12_007F | 120 | Reserved |

1. Duplicate value used if primary value generates a double bit fault when read during the reset sequence.

## 27.3.2 Register Descriptions

The Flash module contains a set of 20 control and status registers located between Flash module base + 0x0000 and 0x0013. A summary of the Flash module registers is given in Figure 27-4 with detailed descriptions in the following subsections.

**CAUTION**

Writes to any Flash register must be avoided while a Flash command is active (CCIF=0) to prevent corruption of Flash register contents and Memory Controller behavior.

| Address & Name | | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|
| 0x0000 FCLKDIV | R | FDIVLD | FDIV6 | FDIV5 | FDIV4 | FDIV3 | FDIV2 | FDIV1 | FDIV0 |
| | W | | | | | | | | |
| 0x0001 FSEC | R | KEYEN1 | KEYEN0 | RNV5 | RNV4 | RNV3 | RNV2 | SEC1 | SEC0 |
| | W | | | | | | | | |
| 0x0002 FCCOBIX | R | 0 | 0 | 0 | 0 | 0 | CCOBIX2 | CCOBIX1 | CCOBIX0 |
| | W | | | | | | | | |
| 0x0003 FECCRIX | R | 0 | 0 | 0 | 0 | 0 | ECCRIX2 | ECCRIX1 | ECCRIX0 |
| | W | | | | | | | | |
| 0x0004 FCNFG | R | CCIE | 0 | 0 | IGNSF | 0 | 0 | FDFD | FSFD |
| | W | | | | | | | | |
| 0x0005 FERCNFG | R | ERSERIE | PGMERIE | 0 | EPVIOLIE | ERSVIE1 | ERSVIE0 | DFDIE | SFDIE |
| | W | | | | | | | | |

**Figure 27-4. FTM512K3 Register Summary**

| Address & Name | | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|
| 0x0006 FSTAT | R | CCIF | 0 | ACCERR | FPVIOL | MGBUSY | RSVD | MGSTAT1 | MGSTAT0 |
| | W | | | | | | | | |
| 0x0007 FERSTAT | R | ERSERIF | PGMERIF | 0 | EPVIOLIF | ERSVIF1 | ERSVIF0 | DFDIF | SFDIF |
| | W | | | | | | | | |
| 0x0008 FPROT | R | FPOPEN | RNV6 | FPHDIS | FPHS1 | FPHS0 | FPLDIS | FPLS1 | FPLS0 |
| | W | | | | | | | | |
| 0x0009 EPROT | R | EPOPEN | RNV6 | RNV5 | RNV4 | EPDIS | EPS2 | EPS1 | EPS0 |
| | W | | | | | | | | |
| 0x000A FCCOBHI | R | CCOB15 | CCOB14 | CCOB13 | CCOB12 | CCOB11 | CCOB10 | CCOB9 | CCOB8 |
| | W | | | | | | | | |
| 0x000B FCCOBLO | R | CCOB7 | CCOB6 | CCOB5 | CCOB4 | CCOB3 | CCOB2 | CCOB1 | CCOB0 |
| | W | | | | | | | | |
| 0x000C ETAGHI | R | ETAG15 | ETAG14 | ETAG13 | ETAG12 | ETAG11 | ETAG10 | ETAG9 | ETAG8 |
| | W | | | | | | | | |
| 0x000D ETAGLO | R | ETAG7 | ETAG6 | ETAG5 | ETAG4 | ETAG3 | ETAG2 | ETAG1 | ETAG0 |
| | W | | | | | | | | |
| 0x000E FECCRHI | R | ECCR15 | ECCR14 | ECCR13 | ECCR12 | ECCR11 | ECCR10 | ECCR9 | ECCR8 |
| | W | | | | | | | | |
| 0x000F FECCRLO | R | ECCR7 | ECCR6 | ECCR5 | ECCR4 | ECCR3 | ECCR2 | ECCR1 | ECCR0 |
| | W | | | | | | | | |
| 0x0010 FOPT | R | NV7 | NV6 | NV5 | NV4 | NV3 | NV2 | NV1 | NV0 |
| | W | | | | | | | | |
| 0x0011 FRSV0 | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | W | | | | | | | | |
| 0x0012 FRSV1 | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | W | | | | | | | | |
| 0x0013 FRSV2 | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | W | | | | | | | | |

**Figure 27-4. FTM512K3 Register Summary (continued)**

**Figure 27-4. FTM512K3 Register Summary (continued)**

### 27.3.2.1 Flash Clock Divider Register (FCLKDIV)

The FCLKDIV register is used to control timed events in program and erase algorithms.

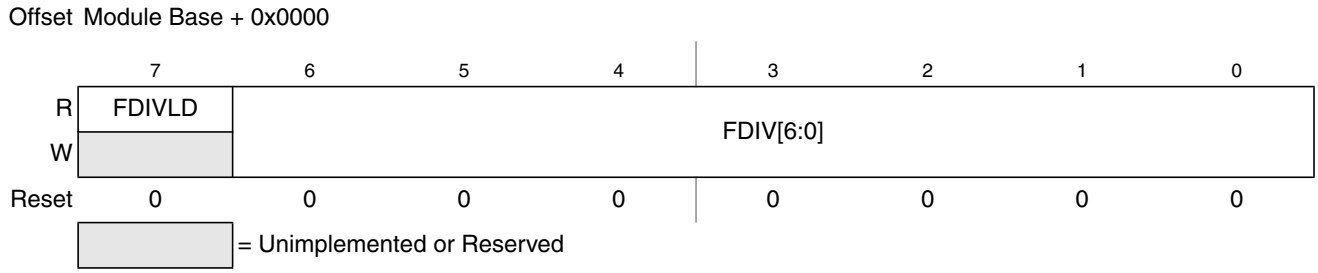



**Figure 27-5. Flash Clock Divider Register (FCLKDIV)**

All bits in the FCLKDIV register are readable, bits 6–0 are write once and bit 7 is not writable.

**Table 27-8. FCLKDIV Field Descriptions**

| Field | Description |
|---|---|
| 7<br>FDIVLD | **Clock Divider Loaded**<br>0 FCLKDIV register has not been written<br>1 FCLKDIV register has been written since the last reset |
| 6–0<br>FDIV[6:0] | **Clock Divider Bits** — FDIV[6:0] must be set to effectively divide OSCCLK down to generate an internal Flash clock, FCLK, with a target frequency of 1 MHz for use by the Flash module to control timed events during program and erase algorithms. Table 27-9 shows recommended values for FDIV[6:0] based on OSCCLK frequency. Please refer to Section 27.4.1, “Flash Command Operations,” for more information. |

**CAUTION**

The FCLKDIV register should never be written while a Flash command is executing (CCIF=0). The FCLKDIV register is writable during the Flash reset sequence even though CCIF is clear.

**Table 27-9. FDIV vs OSCCLK Frequency**

| OSCCLK Frequency (MHz) | | FDIV[6:0] | OSCCLK Frequency (MHz) | | FDIV[6:0] | OSCCLK Frequency (MHz) | | FDIV[6:0] |
|---|---|---|---|---|---|---|---|---|
| MIN(1) | MAX(2) | | MIN1 | MAX2 | | MIN1 | MAX2 | |
| | | | 33.60 | 34.65 | 0x20 | 67.20 | 68.25 | 0x40 |
| 1.60 | 2.10 | 0x01 | 34.65 | 35.70 | 0x21 | 68.25 | 69.30 | 0x41 |
| 2.40 | 3.15 | 0x02 | 35.70 | 36.75 | 0x22 | 69.30 | 70.35 | 0x42 |
| 3.20 | 4.20 | 0x03 | 36.75 | 37.80 | 0x23 | 70.35 | 71.40 | 0x43 |
| 4.20 | 5.25 | 0x04 | 37.80 | 38.85 | 0x24 | 71.40 | 72.45 | 0x44 |
| 5.25 | 6.30 | 0x05 | 38.85 | 39.90 | 0x25 | 72.45 | 73.50 | 0x45 |
| 6.30 | 7.35 | 0x06 | 39.90 | 40.95 | 0x26 | 73.50 | 74.55 | 0x46 |
| 7.35 | 8.40 | 0x07 | 40.95 | 42.00 | 0x27 | 74.55 | 75.60 | 0x47 |
| 8.40 | 9.45 | 0x08 | 42.00 | 43.05 | 0x28 | 75.60 | 76.65 | 0x48 |
| 9.45 | 10.50 | 0x09 | 43.05 | 44.10 | 0x29 | 76.65 | 77.70 | 0x49 |
| 10.50 | 11.55 | 0x0A | 44.10 | 45.15 | 0x2A | 77.70 | 78.75 | 0x4A |
| 11.55 | 12.60 | 0x0B | 45.15 | 46.20 | 0x2B | 78.75 | 79.80 | 0x4B |
| 12.60 | 13.65 | 0x0C | 46.20 | 47.25 | 0x2C | 79.80 | 80.85 | 0x4C |
| 13.65 | 14.70 | 0x0D | 47.25 | 48.30 | 0x2D | 80.85 | 81.90 | 0x4D |
| 14.70 | 15.75 | 0x0E | 48.30 | 49.35 | 0x2E | 81.90 | 82.95 | 0x4E |
| 15.75 | 16.80 | 0x0F | 49.35 | 50.40 | 0x2F | 82.95 | 84.00 | 0x4F |
| 16.80 | 17.85 | 0x10 | 50.40 | 51.45 | 0x30 | 84.00 | 85.05 | 0x50 |
| 17.85 | 18.90 | 0x11 | 51.45 | 52.50 | 0x31 | 85.05 | 86.10 | 0x51 |
| 18.90 | 19.95 | 0x12 | 52.50 | 53.55 | 0x32 | 86.10 | 87.15 | 0x52 |
| 19.95 | 21.00 | 0x13 | 53.55 | 54.60 | 0x33 | 87.15 | 88.20 | 0x53 |
| 21.00 | 22.05 | 0x14 | 54.60 | 55.65 | 0x34 | 88.20 | 89.25 | 0x54 |
| 22.05 | 23.10 | 0x15 | 55.65 | 56.70 | 0x35 | 89.25 | 90.30 | 0x55 |
| 23.10 | 24.15 | 0x16 | 56.70 | 57.75 | 0x36 | 90.30 | 91.35 | 0x56 |
| 24.15 | 25.20 | 0x17 | 57.75 | 58.80 | 0x37 | 91.35 | 92.40 | 0x57 |
| 25.20 | 26.25 | 0x18 | 58.80 | 59.85 | 0x38 | 92.40 | 93.45 | 0x58 |
| 26.25 | 27.30 | 0x19 | 59.85 | 60.90 | 0x39 | 93.45 | 94.50 | 0x59 |
| 27.30 | 28.35 | 0x1A | 60.90 | 61.95 | 0x3A | 94.50 | 95.55 | 0x5A |
| 28.35 | 29.40 | 0x1B | 61.95 | 63.00 | 0x3B | 95.55 | 96.60 | 0x5B |
| 29.40 | 30.45 | 0x1C | 63.00 | 64.05 | 0x3C | 96.60 | 97.65 | 0x5C |
| 30.45 | 31.50 | 0x1D | 64.05 | 65.10 | 0x3D | 97.65 | 98.70 | 0x5D |
| 31.50 | 32.55 | 0x1E | 65.10 | 66.15 | 0x3E | 98.70 | 99.75 | 0x5E |
| 32.55 | 33.60 | 0x1F | 66.15 | 67.20 | 0x3F | 99.75 | 100.80 | 0x5F |

1. FDIV shown generates an FCLK frequency of >0.8 MHz

2. FDIV shown generates an FCLK frequency of 1.05 MHz

### 27.3.2.2 Flash Security Register (FSEC)

The FSEC register holds all bits associated with the security of the MCU and Flash module.

Offset Module Base + 0x0001

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | KEYEN[1:0] | | RNV[5:2] | | | | SEC[1:0] | |
| W | | | | | | | | |
| Reset | F | F | F | F | F | F | F | F |

= Unimplemented or Reserved

**Figure 27-6. Flash Security Register (FSEC)**

All bits in the FSEC register are readable but not writable.

During the reset sequence, the FSEC register is loaded with the contents of the Flash security byte in the Flash configuration field at global address 0x7F_FF0F located in P-Flash memory (see Table 27-3) as indicated by reset condition F in Figure 27-6. If a double bit fault is detected while reading the P-Flash phrase containing the Flash security byte during the reset sequence, all bits in the FSEC register will be set to leave the Flash module in a secured state with backdoor key access disabled.

**Table 27-10. FSEC Field Descriptions**

| Field | Description |
|---|---|
| 7–6<br>KEYEN[1:0] | **Backdoor Key Security Enable Bits** — The KEYEN[1:0] bits define the enabling of backdoor key access to the Flash module as shown in Table 27-11. |
| 5–2<br>RNV[5:2} | **Reserved Nonvolatile Bits** — The RNV bits should remain in the erased state for future enhancements. |
| 1–0<br>SEC[1:0] | **Flash Security Bits** — The SEC[1:0] bits define the security state of the MCU as shown in Table 27-12. If the Flash module is unsecured using backdoor key access, the SEC bits are forced to 10. |

**Table 27-11. Flash KEYEN States**

| KEYEN[1:0] | Status of Backdoor Key Access |
|---|---|
| 00 | DISABLED |
| 01 | DISABLED(1) |
| 10 | ENABLED |
| 11 | DISABLED |

1. Preferred KEYEN state to disable backdoor key access.

**Table 27-12. Flash Security States**

| SEC[1:0] | Status of Security |
|---|---|
| 00 | SECURED |
| 01 | SECURED(1) |
| 10 | UNSECURED |
| 11 | SECURED |

1. Preferred SEC state to set MCU to secured state.

The security function in the Flash module is described in Section 27.5.

### 27.3.2.3 Flash CCOB Index Register (FCCOBIX)

The FCCOBIX register is used to index the FCCOB register for Flash memory operations.

Offset Module Base + 0x0002

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | 0 | 0 | 0 | CCOBIX[2:0] | | |
| W | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented or Reserved

**Figure 27-7. FCCOB Index Register (FCCOBIX)**

CCOBIX bits are readable and writable while remaining bits read 0 and are not writable.

**Table 27-13. FCCOBIX Field Descriptions**

| Field | Description |
|---|---|
| 2–0 CCOBIX[1:0] | **Common Command Register Index**— The CCOBIX bits are used to select which word of the FCCOB register array is being read or written to. See Section 27.3.2.11, “Flash Common Command Object Register (FCCOB),” for more details. |

### 27.3.2.4 Flash ECCR Index Register (FECCRIX)

The FECCRIX register is used to index the FECCR register for ECC fault reporting.

Offset Module Base + 0x0003

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | 0 | 0 | 0 | ECCRIX[2:0] | | |
| W | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented or Reserved

**Figure 27-8. FECCR Index Register (FECCRIX)**

ECCRIX bits are readable and writable while remaining bits read 0 and are not writable.

**Table 27-14. FECCRIX Field Descriptions**

| Field | Description |
|---|---|
| 2-0<br>ECCRIX[2:0] | **ECC Error Register Index**— The ECCRIX bits are used to select which word of the FECCR register array is being read. See Section 27.3.2.13, “Flash ECC Error Results Register (FECCR),” for more details. |

### 27.3.2.5 Flash Configuration Register (FCNFG)

The FCNFG register enables the Flash command complete interrupt and forces ECC faults on Flash array read access from the CPU or XGATE.

Offset Module Base + 0x0004

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | CCIE | 0 | 0 | IGNSF | 0 | 0 | FDFD | FSFD |
| W | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented or Reserved

**Figure 27-9. Flash Configuration Register (FCNFG)**

CCIE, IGNSF, FDFD, and FSFD bits are readable and writable while remaining bits read 0 and are not writable.

**Table 27-15. FCNFG Field Descriptions**

| Field | Description |
|---|---|
| 7<br>CCIE | **Command Complete Interrupt Enable** — The CCIE bit controls interrupt generation when a Flash command has completed.<br>0 Command complete interrupt disabled<br>1 An interrupt will be requested whenever the CCIF flag in the FSTAT register is set (see Section 27.3.2.7) |
| 4<br>IGNSF | **Ignore Single Bit Fault** — The IGNSF controls single bit fault reporting in the FERSTAT register (see Section 27.3.2.8).<br>0 All single bit faults detected during array reads are reported<br>1 Single bit faults detected during array reads are not reported and the single bit fault interrupt will not be generated |

**Table 27-15. FCNFG Field Descriptions (continued)**

| Field | Description |
|---|---|
| 1<br>FDFD | **Force Double Bit Fault Detect** — The FDFD bit allows the user to simulate a double bit fault during Flash array read operations and check the associated interrupt routine. The FDFD bit is cleared by writing a 0 to FDFD. The FECCR registers will not be updated during the Flash array read operation with FDFD set unless an actual double bit fault is detected.<br>0 Flash array read operations will set the DFDIF flag in the FERSTAT register only if a double bit fault is detected<br>1 Any Flash array read operation will force the DFDIF flag in the FERSTAT register to be set (see Section 27.3.2.7) and an interrupt will be generated as long as the DFDIE interrupt enable in the FERCNFG register is set (see Section 27.3.2.6) |
| 0<br>FSFD | **Force Single Bit Fault Detect** — The FSFD bit allows the user to simulate a single bit fault during Flash array read operations and check the associated interrupt routine. The FSFD bit is cleared by writing a 0 to FSFD. The FECCR registers will not be updated during the Flash array read operation with FSFD set unless an actual single bit fault is detected.<br>0 Flash array read operations will set the SFDIF flag in the FERSTAT register only if a single bit fault is detected<br>1 Flash array read operation will force the SFDIF flag in the FERSTAT register to be set (see Section 27.3.2.7) and an interrupt will be generated as long as the SFDIE interrupt enable in the FERCNFG register is set (see Section 27.3.2.6) |

## 27.3.2.6 Flash Error Configuration Register (FERCNFG)

The FERCNFG register enables the Flash error interrupts for the FERSTAT flags.

Offset Module Base + 0x0005

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | ERSERIE | PGMERIE | 0 | EPVIOLIE | ERSVIE1 | ERSVIE0 | DFDIE | SFDIE |
| W | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented or Reserved

**Figure 27-10. Flash Error Configuration Register (FERCNFG)**

All assigned bits in the FERCNFG register are readable and writable.

**Table 27-16. FERCNFG Field Descriptions**

| Field | Description |
|---|---|
| 7<br>ERSERIE | **EEE Erase Error Interrupt Enable** — The ERSERIE bit controls interrupt generation when a failure is detected during an EEE erase operation.<br>0 ERSERIF interrupt disabled<br>1 An interrupt will be requested whenever the ERSERIF flag is set (see Section 27.3.2.8) |
| 6<br>PGMERIE | **EEE Program Error Interrupt Enable** — The PGMERIE bit controls interrupt generation when a failure is detected during an EEE program operation.<br>0 PGMERIF interrupt disabled<br>1 An interrupt will be requested whenever the PGMERIF flag is set (see Section 27.3.2.8) |
| 4<br>EPVIOLIE | **EEE Protection Violation Interrupt Enable** — The EPVIOLIE bit controls interrupt generation when a protection violation is detected during a write to the buffer RAM EEE partition.<br>0 EPVIOLIF interrupt disabled<br>1 An interrupt will be requested whenever the EPVIOLIF flag is set (see Section 27.3.2.8) |

**Table 27-16. FERCNFG Field Descriptions (continued)**

| Field | Description |
|---|---|
| 3<br>ERSVIE1 | **EEE Error Type 1 Interrupt Enable** — The ERSVIE1 bit controls interrupt generation when a change state error is detected during an EEE operation.<br>0 ERSVIF1 interrupt disabled<br>1 An interrupt will be requested whenever the ERSVIF1 flag is set (see Section 27.3.2.8) |
| 2<br>ERSVIE0 | **EEE Error Type 0 Interrupt Enable** — The ERSVIE0 bit controls interrupt generation when a sector format error is detected during an EEE operation.<br>0 ERSVIF0 interrupt disabled<br>1 An interrupt will be requested whenever the ERSVIF0 flag is set (see Section 27.3.2.8) |
| 1<br>DFDIE | **Double Bit Fault Detect Interrupt Enable** — The DFDIE bit controls interrupt generation when a double bit fault is detected during a Flash block read operation.<br>0 DFDIF interrupt disabled<br>1 An interrupt will be requested whenever the DFDIF flag is set (see Section 27.3.2.8) |
| 0<br>SFDIE | **Single Bit Fault Detect Interrupt Enable** — The SFDIE bit controls interrupt generation when a single bit fault is detected during a Flash block read operation.<br>0 SFDIF interrupt disabled whenever the SFDIF flag is set (see Section 27.3.2.8)<br>1 An interrupt will be requested whenever the SFDIF flag is set (see Section 27.3.2.8) |

### 27.3.2.7 Flash Status Register (FSTAT)

The FSTAT register reports the operational status of the Flash module.

Offset Module Base + 0x0006

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | CCIF | 0 | ACCERR | FPVIOL | MGBUSY | RSVD | MGSTAT[1:0] | |
| W | | | | | | | | |
| Reset | 1 | 0 | 0 | 0 | 0 | 0 | 0(1) | 0[1] |

= Unimplemented or Reserved

**Figure 27-11. Flash Status Register (FSTAT)**

1. Reset value can deviate from the value shown if a double bit fault is detected during the reset sequence (see Section 27.6).

CCIF, ACCERR, and FPVIOL bits are readable and writable, MGBUSY and MGSTAT bits are readable but not writable, while remaining bits read 0 and are not writable.

**Table 27-17. FSTAT Field Descriptions**

| Field | Description |
|---|---|
| 7<br>CCIF | **Command Complete Interrupt Flag** — The CCIF flag indicates that a Flash command has completed. The CCIF flag is cleared by writing a 1 to CCIF to launch a command and CCIF will stay low until command completion or command violation.<br>0 Flash command in progress<br>1 Flash command has completed |
| 5<br>ACCERR | **Flash Access Error Flag** — The ACCERR bit indicates an illegal access has occurred to the Flash memory caused by either a violation of the command write sequence (see Section 27.4.1.2) or issuing an illegal Flash command or when errors are encountered while initializing the EEE buffer ram during the reset sequence. While ACCERR is set, the CCIF flag cannot be cleared to launch a command. The ACCERR bit is cleared by writing a 1 to ACCERR. Writing a 0 to the ACCERR bit has no effect on ACCERR.<br>0 No access error detected<br>1 Access error detected |
| 4<br>FPVIOL | **Flash Protection Violation Flag** —The FPVIOL bit indicates an attempt was made to program or erase an address in a protected area of P-Flash memory during a command write sequence. The FPVIOL bit is cleared by writing a 1 to FPVIOL. Writing a 0 to the FPVIOL bit has no effect on FPVIOL. While FPVIOL is set, it is not possible to launch a command or start a command write sequence.<br>0 No protection violation detected<br>1 Protection violation detected |
| 3<br>MGBUSY | **Memory Controller Busy Flag** — The MGBUSY flag reflects the active state of the Memory Controller.<br>0 Memory Controller is idle<br>1 Memory Controller is busy executing a Flash command (CCIF = 0) or is handling internal EEE operations |
| 2<br>RSVD | **Reserved Bit** — This bit is reserved and always reads 0. |
| 1–0<br>MGSTAT[1:0] | **Memory Controller Command Completion Status Flag** — One or more MGSTAT flag bits are set if an error is detected during execution of a Flash command or during the Flash reset sequence. See Section 27.4.2, “Flash Command Description,” and Section 27.6, “Initialization” for details. |

### 27.3.2.8 Flash Error Status Register (FERSTAT)

The FERSTAT register reflects the error status of internal Flash operations.

Offset Module Base + 0x0007

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | ERSERIF | PGMERIF | 0 | EPVIOLIF | ERSVIF1 | ERSVIF0 | DFDIF | SFDIF |
| W | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented or Reserved

**Figure 27-12. Flash Error Status Register (FERSTAT)**

All flags in the FERSTAT register are readable and only writable to clear the flag.

**Table 27-18. FERSTAT Field Descriptions**

| Field | Description |
|---|---|
| 7<br>ERSERIF | **EEE Erase Error Interrupt Flag** — The setting of the ERSERIF flag occurs due to an error in a Flash erase command that resulted in the erase operation not being successful during EEE operations. The ERSERIF flag is cleared by writing a 1 to ERSERIF. Writing a 0 to the ERSERIF flag has no effect on ERSERIF. While ERSERIF is set, it is possible to write to the buffer RAM EEE partition but the data written will not be transferred to the D-Flash EEE partition.<br>0 Erase command successfully completed on the D-Flash EEE partition<br>1 Erase command failed on the D-Flash EEE partition |
| 6<br>PGMERIF | **EEE Program Error Interrupt Flag** — The setting of the PGMERIF flag occurs due to an error in a Flash program command that resulted in the program operation not being successful during EEE operations. The PGMERIF flag is cleared by writing a 1 to PGMERIF. Writing a 0 to the PGMERIF flag has no effect on PGMERIF. While PGMERIF is set, it is possible to write to the buffer RAM EEE partition but the data written will not be transferred to the D-Flash EEE partition.<br>0 Program command successfully completed on the D-Flash EEE partition<br>1 Program command failed on the D-Flash EEE partition |
| 4<br>EPVIOLIF | **EEE Protection Violation Interrupt Flag** —The setting of the EPVIOLIF flag indicates an attempt was made to write to a protected area of the buffer RAM EEE partition. The EPVIOLIF flag is cleared by writing a 1 to EPVIOLIF. Writing a 0 to the EPVIOLIF flag has no effect on EPVIOLIF. While EPVIOLIF is set, it is possible to write to the buffer RAM EEE partition as long as the address written to is not in a protected area.<br>0 No EEE protection violation<br>1 EEE protection violation detected |
| 3<br>ERSVIF1 | **EEE Error Interrupt 1 Flag** —The setting of the ERSVIF1 flag indicates that the memory controller was unable to change the state of a D-Flash EEE sector. The ERSVIF1 flag is cleared by writing a 1 to ERSVIF1. Writing a 0 to the ERSVIF1 flag has no effect on ERSVIF1. While ERSVIF1 is set, it is possible to write to the buffer RAM EEE partition but the data written will not be transferred to the D-Flash EEE partition.<br>0 No EEE sector state change error detected<br>1 EEE sector state change error detected |
| 2<br>ERSVIF0 | **EEE Error Interrupt 0 Flag** —The setting of the ERSVIF0 flag indicates that the memory controller was unable to format a D-Flash EEE sector for EEE use. The ERSVIF0 flag is cleared by writing a 1 to ERSVIF0. Writing a 0 to the ERSVIF0 flag has no effect on ERSVIF0. While ERSVIF0 is set, it is possible to write to the buffer RAM EEE partition but the data written will not be transferred to the D-Flash EEE partition.<br>0 No EEE sector format error detected<br>1 EEE sector format error detected |
| 1<br>DFDIF | **Double Bit Fault Detect Interrupt Flag** — The setting of the DFDIF flag indicates that a double bit fault was detected in the stored parity and data bits during a Flash array read operation or that a Flash array read operation was attempted on a Flash block that was under a Flash command operation. The DFDIF flag is cleared by writing a 1 to DFDIF. Writing a 0 to DFDIF has no effect on DFDIF.<br>0 No double bit fault detected<br>1 Double bit fault detected or an invalid Flash array read operation attempted |
| 0<br>SFDIF | **Single Bit Fault Detect Interrupt Flag** — With the IGNSF bit in the FCNFG register clear, the SFDIF flag indicates that a single bit fault was detected in the stored parity and data bits during a Flash array read operation or that a Flash array read operation was attempted on a Flash block that was under a Flash command operation. The SFDIF flag is cleared by writing a 1 to SFDIF. Writing a 0 to SFDIF has no effect on SFDIF.<br>0 No single bit fault detected<br>1 Single bit fault detected and corrected or an invalid Flash array read operation attempted |

### 27.3.2.9 P-Flash Protection Register (FPROT)

The FPROT register defines which P-Flash sectors are protected against program and erase operations.

Offset Module Base + 0x0008

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | FPOPEN | RNV6 | FPHDIS | FPHS[1:0] | | FPLDIS | FPLS[1:0] | |
| W | | | | | | | | |
| Reset | F | F | F | F | F | F | F | F |

= Unimplemented or Reserved

**Figure 27-13. Flash Protection Register (FPROT)**

The (unreserved) bits of the FPROT register are writable with the restriction that the size of the protected region can only be increased (see Section 27.3.2.9.1, “P-Flash Protection Restrictions,” and Table 27-23).

During the reset sequence, the FPROT register is loaded with the contents of the P-Flash protection byte in the Flash configuration field at global address 0x7F_FF0C located in P-Flash memory (see Table 27-3) as indicated by reset condition ‘F’ in Figure 27-13. To change the P-Flash protection that will be loaded during the reset sequence, the upper sector of the P-Flash memory must be unprotected, then the P-Flash protection byte must be reprogrammed. If a double bit fault is detected while reading the P-Flash phrase containing the P-Flash protection byte during the reset sequence, the FPOPEN bit will be cleared and remaining bits in the FPROT register will be set to leave the P-Flash memory fully protected.

Trying to alter data in any protected area in the P-Flash memory will result in a protection violation error and the FPVIOL bit will be set in the FSTAT register. The block erase of a P-Flash block is not possible if any of the P-Flash sectors contained in the same P-Flash block are protected.

**Table 27-19. FPROT Field Descriptions**

| Field | Description |
|---|---|
| 7<br>FPOPEN | **Flash Protection Operation Enable** — The FPOPEN bit determines the protection function for program or erase operations as shown in Table 27-20 for the P-Flash block.<br>0 When FPOPEN is clear, the FPHDIS and FPLDIS bits define unprotected address ranges as specified by the corresponding FPHS and FPLS bits<br>1 When FPOPEN is set, the FPHDIS and FPLDIS bits enable protection for the address range specified by the corresponding FPHS and FPLS bits |
| 6<br>RNV[6] | **Reserved Nonvolatile Bit** — The RNV bit should remain in the erased state for future enhancements. |
| 5<br>FPHDIS | **Flash Protection Higher Address Range Disable** — The FPHDIS bit determines whether there is a protected/unprotected area in a specific region of the P-Flash memory ending with global address 0x7F_FFFF.<br>0 Protection/Unprotection enabled<br>1 Protection/Unprotection disabled |
| 4–3<br>FPHS[1:0] | **Flash Protection Higher Address Size** — The FPHS bits determine the size of the protected/unprotected area in P-Flash memory as shown inTable 27-21. The FPHS bits can only be written to while the FPHDIS bit is set. |
| 2<br>FPLDIS | **Flash Protection Lower Address Range Disable** — The FPLDIS bit determines whether there is a protected/unprotected area in a specific region of the P-Flash memory beginning with global address 0x7F_8000.<br>0 Protection/Unprotection enabled<br>1 Protection/Unprotection disabled |
| 1–0<br>FPLS[1:0] | **Flash Protection Lower Address Size** — The FPLS bits determine the size of the protected/unprotected area in P-Flash memory as shown in Table 27-22. The FPLS bits can only be written to while the FPLDIS bit is set. |

**Table 27-20. P-Flash Protection Function**

| FPOPEN | FPHDIS | FPLDIS | Function[1] |
|---|---|---|---|
| 1 | 1 | 1 | No P-Flash Protection |
| 1 | 1 | 0 | Protected Low Range |
| 1 | 0 | 1 | Protected High Range |
| 1 | 0 | 0 | Protected High and Low Ranges |
| 0 | 1 | 1 | Full P-Flash Memory Protected |
| 0 | 1 | 0 | Unprotected Low Range |
| 0 | 0 | 1 | Unprotected High Range |
| 0 | 0 | 0 | Unprotected High and Low Ranges |

1. For range sizes, refer to Table 27-21 and Table 27-22.

**Table 27-21. P-Flash Protection Higher Address Range**

| FPHS[1:0] | Global Address Range | Protected Size |
|---|---|---|
| 00 | 0x7F_F800–0x7F_FFFF | 2 Kbytes |
| 01 | 0x7F_F000–0x7F_FFFF | 4 Kbytes |
| 10 | 0x7F_E000–0x7F_FFFF | 8 Kbytes |
| 11 | 0x7F_C000–0x7F_FFFF | 16 Kbytes |

**Table 27-22. P-Flash Protection Lower Address Range**

| FPLS[1:0] | Global Address Range | Protected Size |
|---|---|---|
| 00 | 0x7F_8000–0x7F_83FF | 1 Kbyte |
| 01 | 0x7F_8000–0x7F_87FF | 2 Kbytes |
| 10 | 0x7F_8000–0x7F_8FFF | 4 Kbytes |
| 11 | 0x7F_8000–0x7F_9FFF | 8 Kbytes |

All possible P-Flash protection scenarios are shown in Figure 27-14. Although the protection scheme is loaded from the Flash memory at global address 0x7F_FF0C during the reset sequence, it can be changed by the user. The P-Flash protection scheme can be used by applications requiring reprogramming in single chip mode while providing as much protection as possible if reprogramming is not required.

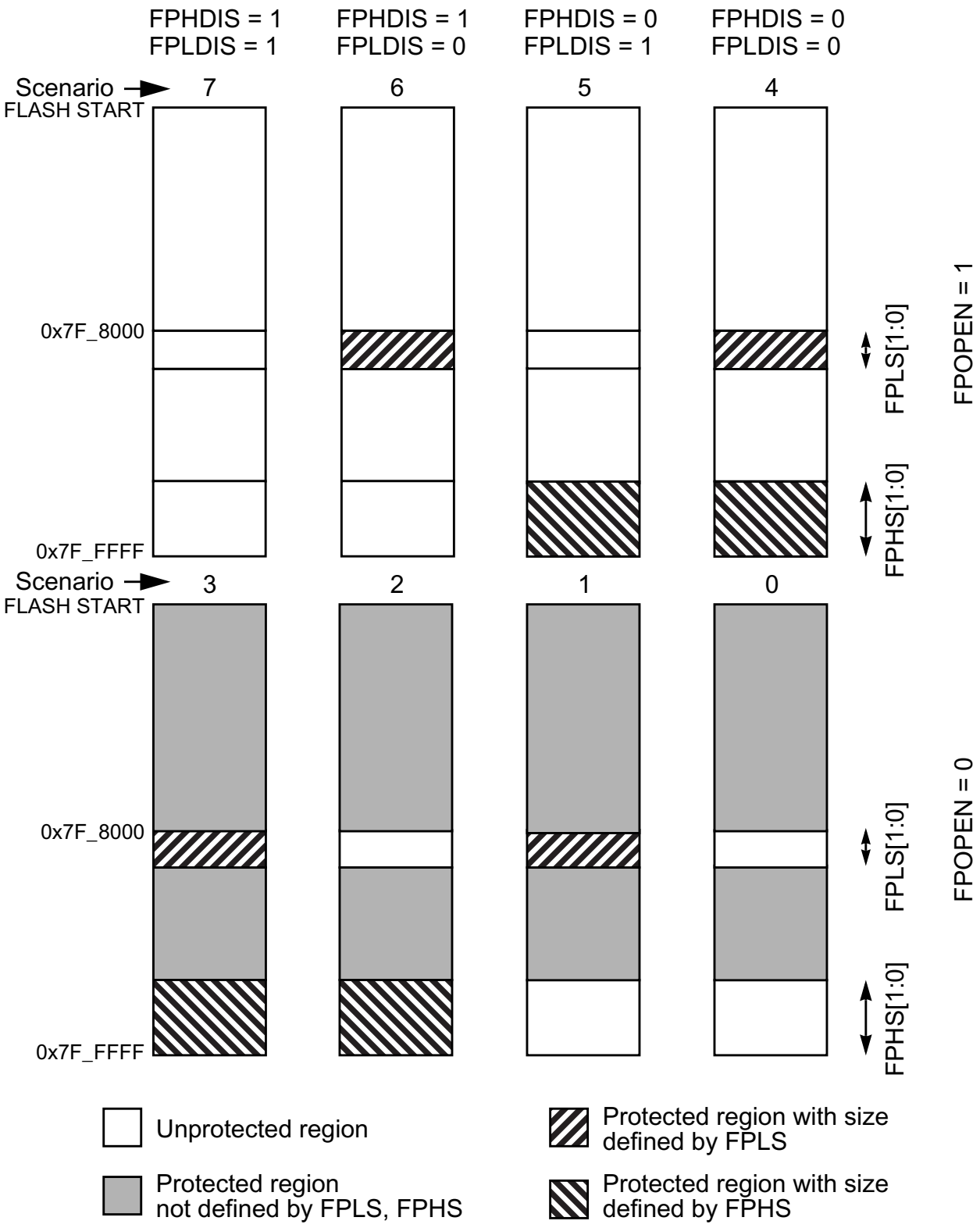



**Figure 27-14. P-Flash Protection Scenarios**

#### 27.3.2.9.1 P-Flash Protection Restrictions

The general guideline is that P-Flash protection can only be added and not removed. Table 27-23 specifies all valid transitions between P-Flash protection scenarios. Any attempt to write an invalid scenario to the FPROT register will be ignored. The contents of the FPROT register reflect the active protection scenario. See the FPHS and FPLS bit descriptions for additional restrictions.

**Table 27-23. P-Flash Protection Scenario Transitions**

| From Protection Scenario | To Protection Scenario[1] | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| 0 | X | X | X | X | | | | |
| 1 | | X | | X | | | | |
| 2 | | | X | X | | | | |
| 3 | | | | X | | | | |
| 4 | | | | X | X | | | |
| 5 | | | X | X | X | X | | |
| 6 | | X | | X | X | | X | |
| 7 | X | X | X | X | X | X | X | X |

1. Allowed transitions marked with X, see Figure 27-14 for a definition of the scenarios.

### 27.3.2.10 EEE Protection Register (EPROT)

The EPROT register defines which buffer RAM EEE partition areas are protected against writes.

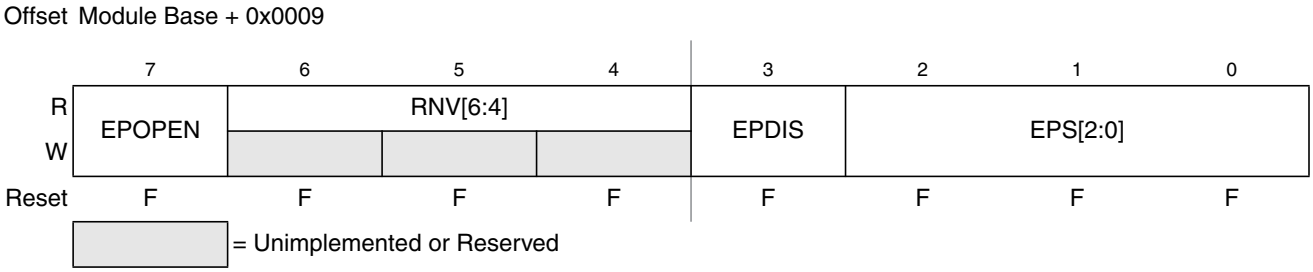



**Figure 27-15. EEE Protection Register (EPROT)**

All bits in the EPROT register are readable and writable except for RNV[6:4] which are only readable. The EPOPEN and EPDIS bits can only be written to the protected state. The EPS bits can be written anytime until the EPDIS bit is cleared. If the EPOPEN bit is cleared, the state of the EPDIS and EPS bits is irrelevant.

During the reset sequence, the EPROT register is loaded from the EEE protection byte in the Flash configuration field at global address 0x7F_FF0D located in P-Flash memory (see Table 27-3) as indicated by reset condition F in Figure 27-15. To change the EEE protection that will be loaded during the reset sequence, the P-Flash sector containing the EEE protection byte must be unprotected, then the EEE protection byte must be programmed. If a double bit fault is detected while reading the P-Flash phrase

containing the EEE protection byte during the reset sequence, the EPOPEN bit will be cleared and remaining bits in the EPROT register will be set to leave the buffer RAM EEE partition fully protected.

Trying to write data to any protected area in the buffer RAM EEE partition will result in a protection violation error and the EPVIOLIF flag will be set in the FERSTAT register. Trying to write data to any protected area in the buffer RAM partitioned for user access will not be prevented and the EPVIOLIF flag in the FERSTAT register will not set.

**Table 27-24. EPROT Field Descriptions**

| Field | Description |
|---|---|
| 7<br>EPOPEN | **Enables writes to the Buffer RAM partitioned for EEE**<br>0 The entire buffer RAM EEE partition is protected from writes<br>1 Unprotected buffer RAM EEE partition areas are enabled for writes |
| 6–4<br>RNV[6:4] | **Reserved Nonvolatile Bits** — The RNV bits should remain in the erased state for future enhancements |
| 3<br>EPDIS | **Buffer RAM Protection Address Range Disable** — The EPDIS bit determines whether there is a protected area in a specific region of the buffer RAM EEE partition.<br>0 Protection enabled<br>1 Protection disabled |
| 2–0<br>EPS[2:0] | **Buffer RAM Protection Size** — The EPS[2:0] bits determine the size of the protected area in the buffer RAM EEE partition as shown inTable 27-21. The EPS bits can only be written to while the EPDIS bit is set. |

**Table 27-25. Buffer RAM EEE Partition Protection Address Range**

| EPS[2:0] | Global Address Range | Protected Size |
|---|---|---|
| 000 | 0x13_FFC0 – 0x13_FFFF | 64 bytes |
| 001 | 0x13_FF80 – 0x13_FFFF | 128 bytes |
| 010 | 0x13_FF40 – 0x13_FFFF | 192 bytes |
| 011 | 0x13_FF00 – 0x13_FFFF | 256 bytes |
| 100 | 0x13_FEC0 – 0x13_FFFF | 320 bytes |
| 101 | 0x13_FE80 – 0x13_FFFF | 384 bytes |
| 110 | 0x13_FE40 – 0x13_FFFF | 448 bytes |
| 111 | 0x13_FE00 – 0x13_FFFF | 512 bytes |

### 27.3.2.11 Flash Common Command Object Register (FCCOB)

The FCCOB is an array of six words addressed via the CCOBIX index found in the FCCOBIX register. Byte wide reads and writes are allowed to the FCCOB register.

Offset Module Base + 0x000A

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | CCOB[15:8] | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 27-16. Flash Common Command Object High Register (FCCOBHI)**

Offset Module Base + 0x000B

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | CCOB[7:0] | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 27-17. Flash Common Command Object Low Register (FCCOBLO)**

### 27.3.2.11.1 FCCOB - NVM Command Mode

NVM command mode uses the indexed FCCOB register to provide a command code and its relevant parameters to the Memory Controller. The user first sets up all required FCCOB fields and then initiates the command's execution by writing a 1 to the CCIF bit in the FSTAT register (a 1 written by the user clears the CCIF command completion flag to 0). When the user clears the CCIF bit in the FSTAT register all FCCOB parameter fields are locked and cannot be changed by the user until the command completes (as evidenced by the Memory Controller returning CCIF to 1). Some commands return information to the FCCOB register array.

The generic format for the FCCOB parameter fields in NVM command mode is shown in Table 27-26. The return values are available for reading after the CCIF flag in the FSTAT register has been returned to 1 by the Memory Controller. Writes to the unimplemented parameter fields (CCOBIX = 110 and CCOBIX = 111) are ignored with reads from these fields returning 0x0000.

Table 27-26 shows the generic Flash command format. The high byte of the first word in the CCOB array contains the command code, followed by the parameters for this specific Flash command. For details on the FCCOB settings required by each command, see the Flash command descriptions in Section 27.4.2.

**Table 27-26. FCCOB - NVM Command Mode (Typical Usage)**

| CCOBIX[2:0] | Byte | FCCOB Parameter Fields (NVM Command Mode) |
|---|---|---|
| 000 | HI | FCMD[7:0] defining Flash command |
| | LO | 0, Global address [22:16] |
| 001 | HI | Global address [15:8] |
| | LO | Global address [7:0] |
| 010 | HI | Data 0 [15:8] |
| | LO | Data 0 [7:0] |

**Table 27-26. FCCOB - NVM Command Mode (Typical Usage)**

| CCOBIX[2:0] | Byte | FCCOB Parameter Fields (NVM Command Mode) |
|---|---|---|
| 011 | HI | Data 1 [15:8] |
| | LO | Data 1 [7:0] |
| 100 | HI | Data 2 [15:8] |
| | LO | Data 2 [7:0] |
| 101 | HI | Data 3 [15:8] |
| | LO | Data 3 [7:0] |

### 27.3.2.12 EEE Tag Counter Register (ETAG)

The ETAG register contains the number of outstanding words in the buffer RAM EEE partition that need to be programmed into the D-Flash EEE partition. The ETAG register is decremented prior to the related tagged word being programmed into the D-Flash EEE partition. All tagged words have been programmed into the D-Flash EEE partition once all bits in the ETAG register read 0 and the MGBUSY flag in the FSTAT register reads 0.

Offset Module Base + 0x000C

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | ETAG[15:8] | | | | | | | |
| W | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented or Reserved

**Figure 27-18. EEE Tag Counter High Register (ETAGHI)**

Offset Module Base + 0x000D

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | ETAG[7:0] | | | | | | | |
| W | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented or Reserved

**Figure 27-19. EEE Tag Counter Low Register (ETAGLO)**

All ETAG bits are readable but not writable and are cleared by the Memory Controller.

### 27.3.2.13 Flash ECC Error Results Register (FECCR)

The FECCR registers contain the result of a detected ECC fault for both single bit and double bit faults. The FECCR register provides access to several ECC related fields as defined by the ECCRIX index bits in the FECCRIX register (see Section 27.3.2.4). Once ECC fault information has been stored, no other

fault information will be recorded until the specific ECC fault flag has been cleared. In the event of simultaneous ECC faults, the priority for fault recording is:

1. Double bit fault over single bit fault
2. CPU over XGATE

Offset Module Base + 0x000E

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | ECCR[15:8] | | | | | | | |
| W | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented or Reserved

**Figure 27-20. Flash ECC Error Results High Register (FECCRHI)**

Offset Module Base + 0x000F

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | ECCR[7:0] | | | | | | | |
| W | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented or Reserved

**Figure 27-21. Flash ECC Error Results Low Register (FECCRLO)**

All FECCR bits are readable but not writable.

**Table 27-27. FECCR Index Settings**

| ECCRIX[2:0] | FECCR Register Content | | |
|---|---|---|---|
| | **Bits [15:8]** | **Bit[7]** | **Bits[6:0]** |
| 000 | Parity bits read from Flash block | CPU or XGATE source identity | Global address [22:16] |
| 001 | Global address [15:0] | | |
| 010 | Data 0 [15:0] | | |
| 011 | Data 1 [15:0] (P-Flash only) | | |
| 100 | Data 2 [15:0] (P-Flash only) | | |
| 101 | Data 3 [15:0] (P-Flash only) | | |
| 110 | Not used, returns 0x0000 when read | | |
| 111 | Not used, returns 0x0000 when read | | |

**Table 27-28. FECCR Index=000 Bit Descriptions**

| Field | Description |
|---|---|
| 15:8<br>PAR[7:0] | **ECC Parity Bits** — Contains the 8 parity bits from the 72 bit wide P-Flash data word or the 6 parity bits, allocated to PAR[5:0], from the 22 bit wide D-Flash word with PAR[7:6]=00. |
| 7<br>XBUS01 | **Bus Source Identifier** — The XBUS01 bit determines whether the ECC error was caused by a read access from the CPU or XGATE.<br>0 ECC Error happened on the CPU access<br>1 ECC Error happened on the XGATE access |
| 6–0<br>GADDR[22:16] | **Global Address** — The GADDR[22:16] field contains the upper seven bits of the global address having caused the error. |

The P-Flash word addressed by ECCRIX = 001 contains the lower 16 bits of the global address. The following four words addressed by ECCRIX = 010 to 101 contain the 64-bit wide data phrase. The four data words and the parity byte are the uncorrected data read from the P-Flash block.

The D-Flash word addressed by ECCRIX = 001 contains the lower 16 bits of the global address. The uncorrected 16-bit data word is addressed by ECCRIX = 010.

### 27.3.2.14 Flash Option Register (FOPT)

The FOPT register is the Flash option register.

Offset Module Base + 0x0010

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | NV[7:0] | | | | | | | |
| W | | | | | | | | |
| Reset | F | F | F | F | F | F | F | F |

= Unimplemented or Reserved

**Figure 27-22. Flash Option Register (FOPT)**

All bits in the FOPT register are readable but are not writable.

During the reset sequence, the FOPT register is loaded from the Flash nonvolatile byte in the Flash configuration field at global address 0x7F_FF0E located in P-Flash memory (see Table 27-3) as indicated by reset condition F in Figure 27-22. If a double bit fault is detected while reading the P-Flash phrase containing the Flash nonvolatile byte during the reset sequence, all bits in the FOPT register will be set.

**Table 27-29. FOPT Field Descriptions**

| Field | Description |
|---|---|
| 7–0<br>NV[7:0] | **Nonvolatile Bits** — The NV[7:0] bits are available as nonvolatile bits. Refer to the device user guide for proper use of the NV bits. |

### 27.3.2.15 Flash Reserved0 Register (FRSV0)

This Flash register is reserved for factory testing.

Offset Module Base + 0x0011

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| W | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented or Reserved

**Figure 27-23. Flash Reserved0 Register (FRSV0)**

All bits in the FRSV0 register read 0 and are not writable.

### 27.3.2.16 Flash Reserved1 Register (FRSV1)

This Flash register is reserved for factory testing.

Offset Module Base + 0x0012

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| W | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented or Reserved

**Figure 27-24. Flash Reserved1 Register (FRSV1)**

All bits in the FRSV1 register read 0 and are not writable.

### 27.3.2.17 Flash Reserved2 Register (FRSV2)

This Flash register is reserved for factory testing.

Offset Module Base + 0x0013

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| W | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented or Reserved

**Figure 27-25. Flash Reserved2 Register (FRSV2)**

All bits in the FRSV2 register read 0 and are not writable.

## 27.4 Functional Description

### 27.4.1 Flash Command Operations

Flash command operations are used to modify Flash memory contents or configure module resources for EEE operation.

The next sections describe:

- How to write the FCLKDIV register that is used to generate a time base (FCLK) derived from OSCCLK for Flash program and erase command operations
- The command write sequence used to set Flash command parameters and launch execution
- Valid Flash commands available for execution

#### 27.4.1.1 Writing the FCLKDIV Register

Prior to issuing any Flash program or erase command after a reset, the user is required to write the FCLKDIV register to divide OSCCLK down to a target FCLK of 1 MHz. Table 27-9 shows recommended values for the FDIV field based on OSCCLK frequency.

**NOTE**

Programming or erasing the Flash memory cannot be performed if the bus clock runs at less than 1 MHz. Setting FDIV too high can destroy the Flash memory due to overstress. Setting FDIV too low can result in incomplete programming or erasure of the Flash memory cells.

When the FCLKDIV register is written, the FDIVLD bit is set automatically. If the FDIVLD bit is 0, the FCLKDIV register has not been written since the last reset. If the FCLKDIV register has not been written, any Flash program or erase command loaded during a command write sequence will not execute and the ACCERR bit in the FSTAT register will set.

#### 27.4.1.2 Command Write Sequence

The Memory Controller will launch all valid Flash commands entered using a command write sequence.

Before launching a command, the ACCERR and FPVIOL bits in the FSTAT register must be clear (see Section 27.3.2.7) and the CCIF flag should be tested to determine the status of the current command write sequence. If CCIF is 0, the previous command write sequence is still active, a new command write sequence cannot be started, and all writes to the FCCOB register are ignored.

**CAUTION**

Writes to any Flash register must be avoided while a Flash command is active (CCIF=0) to prevent corruption of Flash register contents and Memory Controller behavior.

#### 27.4.1.2.1 Define FCCOB Contents

The FCCOB parameter fields must be loaded with all required parameters for the Flash command being executed. Access to the FCCOB parameter fields is controlled via the CCOBIX bits in the FCCOBIX register (see Section 27.3.2.3).

The contents of the FCCOB parameter fields are transferred to the Memory Controller when the user clears the CCIF command completion flag in the FSTAT register (writing 1 clears the CCIF to 0). The CCIF flag will remain clear until the Flash command has completed. Upon completion, the Memory Controller will return CCIF to 1 and the FCCOB register will be used to communicate any results. The flow for a generic command write sequence is shown in Figure 27-26.

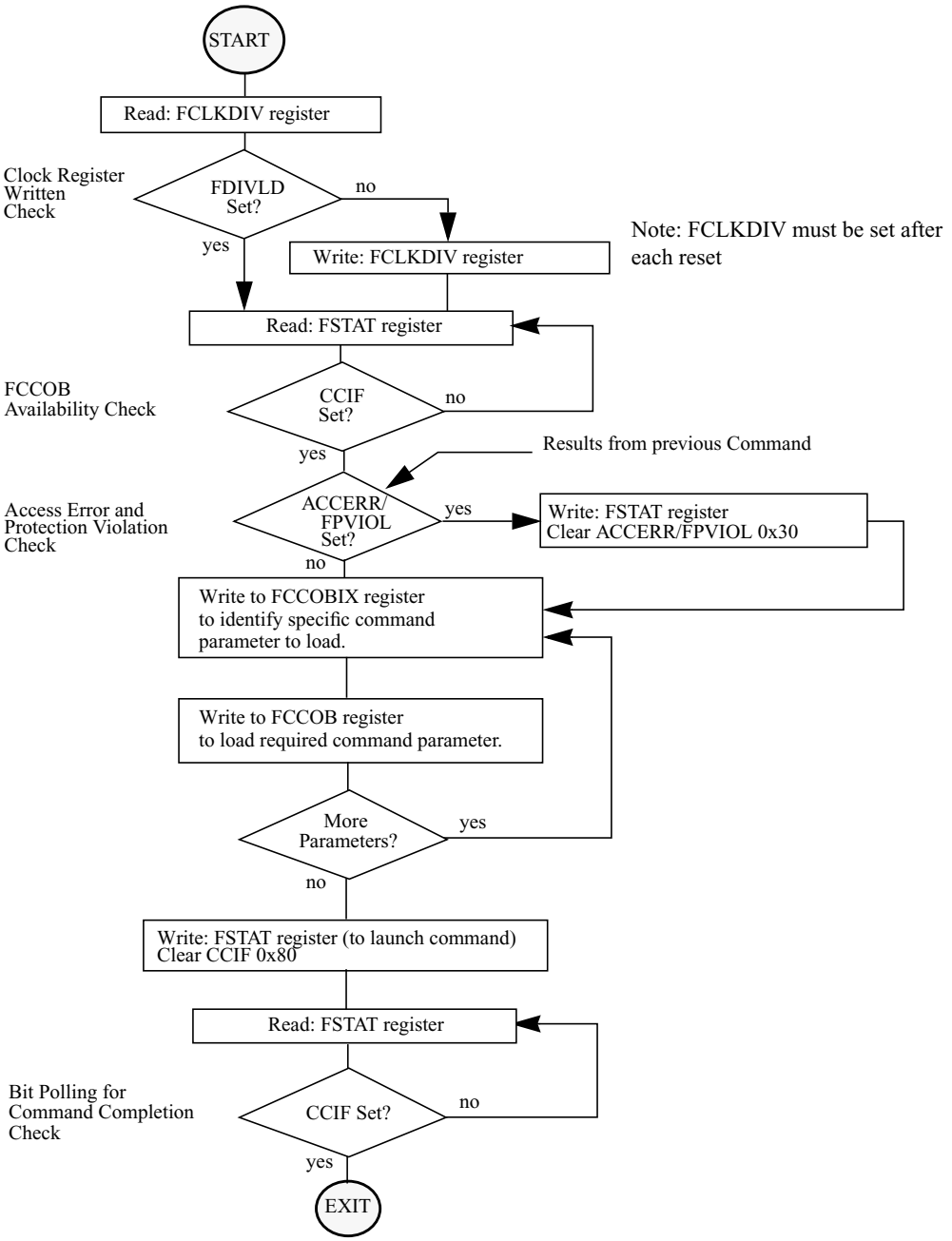



**Figure 27-26. Generic Flash Command Write Sequence Flowchart**

### 27.4.1.3 Valid Flash Module Commands

**Table 27-30. Flash Commands by Mode**

| FCMD | Command | Unsecured | | | | Secured | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | NS (1) | NX (2) | SS(3) | ST(4) | NS (5) | NX (6) | SS(7) | ST(8) |
| 0x01 | Erase Verify All Blocks | * | * | * | * | * | * | * | * |
| 0x02 | Erase Verify Block | * | * | * | * | * | * | * | * |
| 0x03 | Erase Verify P-Flash Section | * | * | * | * | * | | | |
| 0x04 | Read Once | * | * | * | * | * | | | |
| 0x05 | Load Data Field | * | * | * | * | * | | | |
| 0x06 | Program P-Flash | * | * | * | * | * | | | |
| 0x07 | Program Once | * | * | * | * | * | | | |
| 0x08 | Erase All Blocks | | | * | * | | | * | * |
| 0x09 | Erase P-Flash Block | * | * | * | * | * | | | |
| 0x0A | Erase P-Flash Sector | * | * | * | * | * | | | |
| 0x0B | Unsecure Flash | | | * | * | | | * | * |
| 0x0C | Verify Backdoor Access Key | * | | | | * | | | |
| 0x0D | Set User Margin Level | * | * | * | * | * | | | |
| 0x0E | Set Field Margin Level | | | * | * | | | | |
| 0x0F | Full Partition D-Flash | | | * | * | | | | |
| 0x10 | Erase Verify D-Flash Section | * | * | * | * | * | | | |
| 0x11 | Program D-Flash | * | * | * | * | * | | | |
| 0x12 | Erase D-Flash Sector | * | * | * | * | * | | | |
| 0x13 | Enable EEPROM Emulation | * | * | * | * | * | * | * | * |
| 0x14 | Disable EEPROM Emulation | * | * | * | * | * | * | * | * |
| 0x15 | EEPROM Emulation Query | * | * | * | * | * | * | * | * |
| 0x20 | Partition D-Flash | * | * | * | * | * | * | * | * |

1. Unsecured Normal Single Chip mode.
2. Unsecured Normal Expanded mode.
3. Unsecured Special Single Chip mode.
4. Unsecured Special Mode.
5. Secured Normal Single Chip mode.
6. Secured Normal Expanded mode.
7. Secured Special Single Chip mode.
8. Secured Special Mode.

#### 27.4.1.4 P-Flash Commands

Table 27-31 summarizes the valid P-Flash commands along with the effects of the commands on the P-Flash block and other resources within the Flash module.

**Table 27-31. P-Flash Commands**

| FCMD | Command | Function on P-Flash Memory |
|---|---|---|
| 0x01 | Erase Verify All Blocks | Verify that all P-Flash (and D-Flash) blocks are erased. |
| 0x02 | Erase Verify Block | Verify that a P-Flash block is erased. |
| 0x03 | Erase Verify P-Flash Section | Verify that a given number of words starting at the address provided are erased. |
| 0x04 | Read Once | Read a dedicated 64 byte field in the nonvolatile information register in P-Flash block 0 that was previously programmed using the Program Once command. |
| 0x05 | Load Data Field | Load data for simultaneous multiple P-Flash block operations. |
| 0x06 | Program P-Flash | Program a phrase in a P-Flash block and any previously loaded phrases for any other P-Flash block (see Load Data Field command). |
| 0x07 | Program Once | Program a dedicated 64 byte field in the nonvolatile information register in P-Flash block 0 that is allowed to be programmed only once. |
| 0x08 | Erase All Blocks | Erase all P-Flash (and D-Flash) blocks.<br>An erase of all Flash blocks is only possible when the FPLDIS, FPHDIS, and FPOPEN bits in the FPROT register and the EPDIS and EPOPEN bits in the EPROT register are set prior to launching the command. |
| 0x09 | Erase P-Flash Block | Erase a single P-Flash block.<br>An erase of the full P-Flash block is only possible when FPLDIS, FPHDIS and FPOPEN bits in the FPROT register are set prior to launching the command. |
| 0x0A | Erase P-Flash Sector | Erase all bytes in a P-Flash sector. |
| 0x0B | Unsecure Flash | Supports a method of releasing MCU security by erasing all P-Flash (and D-Flash) blocks and verifying that all P-Flash (and D-Flash) blocks are erased. |
| 0x0C | Verify Backdoor Access Key | Supports a method of releasing MCU security by verifying a set of security keys. |
| 0x0D | Set User Margin Level | Specifies a user margin read level for all P-Flash blocks. |
| 0x0E | Set Field Margin Level | Specifies a field margin read level for all P-Flash blocks (special modes only). |

#### 27.4.1.5 D-Flash and EEE Commands

Table 27-32 summarizes the valid D-Flash and EEE commands along with the effects of the commands on the D-Flash block and EEE operation.

**Table 27-32. D-Flash Commands**

| FCMD | Command | Function on D-Flash Memory |
|---|---|---|
| 0x01 | Erase Verify All Blocks | Verify that all D-Flash (and P-Flash) blocks are erased. |
| 0x02 | Erase Verify Block | Verify that the D-Flash block is erased. |

**Table 27-32. D-Flash Commands**

| FCMD | Command | Function on D-Flash Memory |
|---|---|---|
| 0x08 | Erase All Blocks | Erase all D-Flash (and P-Flash) blocks.<br>An erase of all Flash blocks is only possible when the FPLDIS, FPHDIS, and FPOPEN bits in the FPROT register and the EPDIS and EPOPEN bits in the EPROT register are set prior to launching the command. |
| 0x0B | Unsecure Flash | Supports a method of releasing MCU security by erasing all D-Flash (and P-Flash) blocks and verifying that all D-Flash (and P-Flash) blocks are erased. |
| 0x0D | Set User Margin Level | Specifies a user margin read level for the D-Flash block. |
| 0x0E | Set Field Margin Level | Specifies a field margin read level for the D-Flash block (special modes only). |
| 0x0F | Full Partition D-Flash | Erase the D-Flash block and partition an area of the D-Flash block for user access. |
| 0x10 | Erase Verify D-Flash Section | Verify that a given number of words starting at the address provided are erased. |
| 0x11 | Program D-Flash | Program up to four words in the D-Flash block. |
| 0x12 | Erase D-Flash Sector | Erase all bytes in a sector of the D-Flash block. |
| 0x13 | Enable EEPROM Emulation | Enable EEPROM emulation where writes to the buffer RAM EEE partition will be copied to the D-Flash EEE partition. |
| 0x14 | Disable EEPROM Emulation | Suspend all current erase and program activity related to EEPROM emulation but leave current EEE tags set. |
| 0x15 | EEPROM Emulation Query | Returns EEE partition and status variables. |
| 0x20 | Partition D-Flash | Partition an area of the D-Flash block for user access. |

## 27.4.2 Flash Command Description

This section provides details of all available Flash commands launched by a command write sequence. The ACCERR bit in the FSTAT register will be set during the command write sequence if any of the following illegal steps are performed, causing the command not to be processed by the Memory Controller:

- Starting any command write sequence that programs or erases Flash memory before initializing the FCLKDIV register
- Writing an invalid command as part of the command write sequence
- For additional possible errors, refer to the error handling table provided for each command

If a Flash block is read during execution of an algorithm (CCIF = 0) on that same block, the read operation will return invalid data. If the SFDIF or DFDIF flags were not previously set when the invalid read operation occurred, both the SFDIF and DFDIF flags will be set and the FECCR registers will be loaded with the global address used in the invalid read operation with the data and parity fields set to all 0.

If the ACCERR or FPVIOL bits are set in the FSTAT register, the user must clear these bits before starting any command write sequence (see Section 27.3.2.7).

**CAUTION**

A Flash word or phrase must be in the erased state before being programmed. Cumulative programming of bits within a Flash word or phrase is not allowed.

### 27.4.2.1 Erase Verify All Blocks Command

The Erase Verify All Blocks command will verify that all P-Flash and D-Flash blocks have been erased.

**Table 27-33. Erase Verify All Blocks Command FCCOB Requirements**

| CCOBIX[2:0] | FCCOB Parameters | |
|---|---|---|
| 000 | 0x01 | Not required |

Upon clearing CCIF to launch the Erase Verify All Blocks command, the Memory Controller will verify that the entire Flash memory space is erased. The CCIF flag will set after the Erase Verify All Blocks operation has completed.

**Table 27-34. Erase Verify All Blocks Command Error Handling**

| Register | Error Bit | Error Condition |
|---|---|---|
| FSTAT | ACCERR | Set if CCOBIX[2:0] != 000 at command launch |
| | | Set if a Load Data Field command sequence is currently active |
| | FPVIOL | None |
| | MGSTAT1 | Set if any errors have been encountered during the read |
| | MGSTAT0 | Set if any non-correctable errors have been encountered during the read |
| FERSTAT | EPVIOLIF | None |

### 27.4.2.2 Erase Verify Block Command

The Erase Verify Block command allows the user to verify that an entire P-Flash or D-Flash block has been erased. The FCCOB upper global address bits determine which block must be verified.

**Table 27-35. Erase Verify Block Command FCCOB Requirements**

| CCOBIX[2:0] | FCCOB Parameters | |
|---|---|---|
| 000 | 0x02 | Global address [22:16] of the Flash block to be verified. |

Upon clearing CCIF to launch the Erase Verify Block command, the Memory Controller will verify that the selected P-Flash or D-Flash block is erased. The CCIF flag will set after the Erase Verify Block operation has completed.

**Table 27-36. Erase Verify Block Command Error Handling**

| Register | Error Bit | Error Condition |
|---|---|---|
| FSTAT | ACCERR | Set if CCOBIX[2:0] != 000 at command launch |
| | | Set if a Load Data Field command sequence is currently active |
| | | Set if an invalid global address [22:16] is supplied |
| | FPVIOL | None |
| | MGSTAT1 | Set if any errors have been encountered during the read |
| | MGSTAT0 | Set if any non-correctable errors have been encountered during the read |
| FERSTAT | EPVIOLIF | None |

### 27.4.2.3 Erase Verify P-Flash Section Command

The Erase Verify P-Flash Section command will verify that a section of code in the P-Flash memory is erased. The Erase Verify P-Flash Section command defines the starting point of the code to be verified and the number of phrases. The section to be verified cannot cross a 256 Kbyte boundary in the P-Flash memory space.

**Table 27-37. Erase Verify P-Flash Section Command FCCOB Requirements**

| CCOBIX[2:0] | FCCOB Parameters | |
|---|---|---|
| 000 | 0x03 | Global address [22:16] of a P-Flash block |
| 001 | Global address [15:0] of the first phrase to be verified | |
| 010 | Number of phrases to be verified | |

Upon clearing CCIF to launch the Erase Verify P-Flash Section command, the Memory Controller will verify the selected section of Flash memory is erased. The CCIF flag will set after the Erase Verify P-Flash Section operation has completed.

**Table 27-38. Erase Verify P-Flash Section Command Error Handling**

| Register | Error Bit | Error Condition |
|---|---|---|
| FSTAT | ACCERR | Set if CCOBIX[2:0] != 010 at command launch |
| | | Set if a Load Data Field command sequence is currently active |
| | | Set if command not available in current mode (see Table 27-30) |
| | | Set if an invalid global address [22:0] is supplied |
| | | Set if a misaligned phrase address is supplied (global address [2:0] != 000) |
| | | Set if the requested section crosses a 256 Kbyte boundary |
| | FPVIOL | None |
| | MGSTAT1 | Set if any errors have been encountered during the read |
| | MGSTAT0 | Set if any non-correctable errors have been encountered during the read |

**Table 27-38. Erase Verify P-Flash Section Command Error Handling**

| Register | Error Bit | Error Condition |
|---|---|---|
| FERSTAT | EPVIOLIF | None |

### 27.4.2.4 Read Once Command

The Read Once command provides read access to a reserved 64 byte field (8 phrases) located in the nonvolatile information register of P-Flash block 0. The Read Once field is programmed using the Program Once command described in Section 27.4.2.7. The Read Once command must not be executed from the Flash block containing the Program Once reserved field to avoid code runaway.

**Table 27-39. Read Once Command FCCOB Requirements**

| CCOBIX[2:0] | FCCOB Parameters | |
|---|---|---|
| 000 | 0x04 | Not Required |
| 001 | Read Once phrase index (0x0000 - 0x0007) | |
| 010 | Read Once word 0 value | |
| 011 | Read Once word 1 value | |
| 100 | Read Once word 2 value | |
| 101 | Read Once word 3 value | |

Upon clearing CCIF to launch the Read Once command, a Read Once phrase is fetched and stored in the FCCOB indexed register. The CCIF flag will set after the Read Once operation has completed. Valid phrase index values for the Read Once command range from 0x0000 to 0x0007. During execution of the Read Once command, any attempt to read addresses within P-Flash block 0 will return invalid data.

**Table 27-40. Read Once Command Error Handling**

| Register | Error Bit | Error Condition |
|---|---|---|
| FSTAT | ACCERR | Set if CCOBIX[2:0] != 001 at command launch |
| | | Set if a Load Data Field command sequence is currently active |
| | | Set if command not available in current mode (see Table 27-30) |
| | | Set if an invalid phrase index is supplied |
| | FPVIOL | None |
| | MGSTAT1 | Set if any errors have been encountered during the read |
| | MGSTAT0 | Set if any non-correctable errors have been encountered during the read |
| FERSTAT | EPVIOLIF | None |

### 27.4.2.5 Load Data Field Command

The Load Data Field command is executed to provide FCCOB parameters for multiple P-Flash blocks for a future simultaneous program operation in the P-Flash memory space.

**Table 27-41. Load Data Field Command FCCOB Requirements**

| CCOBIX[2:0] | FCCOB Parameters | |
|---|---|---|
| 000 | 0x05 | Global address [22:16] to identify P-Flash block |
| 001 | Global address [15:0] of phrase location to be programmed(1) | |
| 010 | Word 0 | |
| 011 | Word 1 | |
| 100 | Word 2 | |
| 101 | Word 3 | |

1. Global address [2:0] must be 000

Upon clearing CCIF to launch the Load Data Field command, the FCCOB registers will be transferred to the Memory Controller and be programmed in the block specified at the global address given with a future Program P-Flash command launched on a P-Flash block. The CCIF flag will set after the Load Data Field operation has completed. Note that once a Load Data Field command sequence has been initiated, the Load Data Field command sequence will be cancelled if any command other than Load Data Field or the future Program P-Flash is launched. Similarly, if an error occurs after launching a Load Data Field or Program P-Flash command, the associated Load Data Field command sequence will be cancelled.

**Table 27-42. Load Data Field Command Error Handling**

| Register | Error Bit | Error Condition |
|---|---|---|
| FSTAT | ACCERR | Set if CCOBIX[2:0] != 101 at command launch |
| | | Set if command not available in current mode (see Table 27-30) |
| | | Set if an invalid global address [22:0] is supplied |
| | | Set if a misaligned phrase address is supplied (global address [2:0] != 000) |
| | | Set if a Load Data Field command sequence is currently active and the selected block has previously been selected in the same command sequence |
| | | Set if a Load Data Field command sequence is currently active and global address [17:0] does not match that previously supplied in the same command sequence |
| | FPVIOL | Set if the global address [22:0] points to a protected area |
| | MGSTAT1 | None |
| | MGSTAT0 | None |
| FERSTAT | EPVIOLIF | None |

### 27.4.2.6 Program P-Flash Command

The Program P-Flash operation will program a previously erased phrase in the P-Flash memory using an embedded algorithm.

**CAUTION**

A P-Flash phrase must be in the erased state before being programmed. Cumulative programming of bits within a Flash phrase is not allowed.

**Table 27-43. Program P-Flash Command FCCOB Requirements**

| CCOBIX[2:0] | FCCOB Parameters | |
|---|---|---|
| 000 | 0x06 | Global address [22:16] to identify P-Flash block |
| 001 | Global address [15:0] of phrase location to be programmed(1) | |
| 010 | Word 0 program value | |
| 011 | Word 1 program value | |
| 100 | Word 2 program value | |
| 101 | Word 3 program value | |

1. Global address [2:0] must be 000

Upon clearing CCIF to launch the Program P-Flash command, the Memory Controller will program the data words to the supplied global address and will then proceed to verify the data words read back as expected. The CCIF flag will set after the Program P-Flash operation has completed.

**Table 27-44. Program P-Flash Command Error Handling**

| Register | Error Bit | Error Condition |
|---|---|---|
| FSTAT | ACCERR | Set if CCOBIX[2:0] != 101 at command launch |
| | | Set if command not available in current mode (see Table 27-30) |
| | | Set if an invalid global address [22:0] is supplied |
| | | Set if a misaligned phrase address is supplied (global address [2:0] != 000) |
| | | Set if a Load Data Field command sequence is currently active and the selected block has previously been selected in the same command sequence |
| | | Set if a Load Data Field command sequence is currently active and global address [17:0] does not match that previously supplied in the same command sequence |
| | FPVIOL | Set if the global address [22:0] points to a protected area |
| | MGSTAT1 | Set if any errors have been encountered during the verify operation |
| | MGSTAT0 | Set if any non-correctable errors have been encountered during the verify operation |
| FERSTAT | EPVIOLIF | None |

### 27.4.2.7 Program Once Command

The Program Once command restricts programming to a reserved 64 byte field (8 phrases) in the nonvolatile information register located in P-Flash block 0. The Program Once reserved field can be read using the Read Once command as described in Section 27.4.2.4. The Program Once command must only be issued once since the nonvolatile information register in P-Flash block 0 cannot be erased. The Program

Once command must not be executed from the Flash block containing the Program Once reserved field to avoid code runaway.

**Table 27-45. Program Once Command FCCOB Requirements**

| CCOBIX[2:0] | FCCOB Parameters | |
|---|---|---|
| 000 | 0x07 | Not Required |
| 001 | Program Once phrase index (0x0000 - 0x0007) | |
| 010 | Program Once word 0 value | |
| 011 | Program Once word 1 value | |
| 100 | Program Once word 2 value | |
| 101 | Program Once word 3 value | |

Upon clearing CCIF to launch the Program Once command, the Memory Controller first verifies that the selected phrase is erased. If erased, then the selected phrase will be programmed and then verified with read back. The CCIF flag will remain clear, setting only after the Program Once operation has completed.

The reserved nonvolatile information register accessed by the Program Once command cannot be erased and any attempt to program one of these phrases a second time will not be allowed. Valid phrase index values for the Program Once command range from 0x0000 to 0x0007. During execution of the Program Once command, any attempt to read addresses within P-Flash block 0 will return invalid data.

**Table 27-46. Program Once Command Error Handling**

| Register | Error Bit | Error Condition |
|---|---|---|
| FSTAT | ACCERR | Set if CCOBIX[2:0] != 101 at command launch |
| | | Set if a Load Data Field command sequence is currently active |
| | | Set if command not available in current mode (see Table 27-30) |
| | | Set if an invalid phrase index is supplied |
| | | Set if the requested phrase has already been programmed(1) |
| | FPVIOL | None |
| | MGSTAT1 | Set if any errors have been encountered during the verify operation |
| | MGSTAT0 | Set if any non-correctable errors have been encountered during the verify operation |
| FERSTAT | EPVIOLIF | None |

1. If a Program Once phrase is initially programmed to 0xFFFF_FFFF_FFFF_FFFF, the Program Once command will be allowed to execute again on that same phrase.

### 27.4.2.8 Erase All Blocks Command

The Erase All Blocks operation will erase the entire P-Flash and D-Flash memory space including the EEE nonvolatile information register.

**Table 27-47. Erase All Blocks Command FCCOB Requirements**

| CCOBIX[2:0] | FCCOB Parameters | |
|---|---|---|
| 000 | 0x08 | Not required |

Upon clearing CCIF to launch the Erase All Blocks command, the Memory Controller will erase the entire Flash memory space and verify that it is erased. If the Memory Controller verifies that the entire Flash memory space was properly erased, security will be released. During the execution of this command (CCIF=0) the user must not write to any Flash module register. The CCIF flag will set after the Erase All Blocks operation has completed.

**Table 27-48. Erase All Blocks Command Error Handling**

| Register | Error Bit | Error Condition |
|---|---|---|
| FSTAT | ACCERR | Set if CCOBIX[2:0] != 000 at command launch |
| | | Set if a Load Data Field command sequence is currently active |
| | | Set if command not available in current mode (see Table 27-30) |
| | FPVIOL | Set if any area of the P-Flash memory is protected |
| | MGSTAT1 | Set if any errors have been encountered during the verify operation |
| | MGSTAT0 | Set if any non-correctable errors have been encountered during the verify operation |
| FERSTAT | EPVIOLIF | Set if any area of the buffer RAM EEE partition is protected |

### 27.4.2.9 Erase P-Flash Block Command

The Erase P-Flash Block operation will erase all addresses in a P-Flash block.

**Table 27-49. Erase P-Flash Block Command FCCOB Requirements**

| CCOBIX[2:0] | FCCOB Parameters | |
|---|---|---|
| 000 | 0x09 | Global address [22:16] to identify P-Flash block |
| 001 | Global address [15:0] in P-Flash block to be erased | |

Upon clearing CCIF to launch the Erase P-Flash Block command, the Memory Controller will erase the selected P-Flash block and verify that it is erased. The CCIF flag will set after the Erase P-Flash Block operation has completed.

**Table 27-50. Erase P-Flash Block Command Error Handling**

| Register | Error Bit | Error Condition |
|---|---|---|
| FSTAT | ACCERR | Set if CCOBIX[2:0] != 001 at command launch |
| | | Set if a Load Data Field command sequence is currently active |
| | | Set if command not available in current mode (see Table 27-30) |
| | | Set if an invalid global address [22:16] is supplied |
| | FPVIOL | Set if an area of the selected P-Flash block is protected |
| | MGSTAT1 | Set if any errors have been encountered during the verify operation |
| | MGSTAT0 | Set if any non-correctable errors have been encountered during the verify operation |
| FERSTAT | EPVIOLIF | None |

### 27.4.2.10 Erase P-Flash Sector Command

The Erase P-Flash Sector operation will erase all addresses in a P-Flash sector.

**Table 27-51. Erase P-Flash Sector Command FCCOB Requirements**

| CCOBIX[2:0] | FCCOB Parameters | |
|---|---|---|
| 000 | 0x0A | Global address [22:16] to identify P-Flash block to be erased |
| 001 | Global address [15:0] anywhere within the sector to be erased. Refer to Section 27.1.2.1 for the P-Flash sector size. | |

Upon clearing CCIF to launch the Erase P-Flash Sector command, the Memory Controller will erase the selected Flash sector and then verify that it is erased. The CCIF flag will be set after the Erase P-Flash Sector operation has completed.

**Table 27-52. Erase P-Flash Sector Command Error Handling**

| Register | Error Bit | Error Condition |
|---|---|---|
| FSTAT | ACCERR | Set if CCOBIX[2:0] != 001 at command launch |
| | | Set if a Load Data Field command sequence is currently active |
| | | Set if command not available in current mode (see Table 27-30) |
| | | Set if an invalid global address [22:16] is supplied |
| | | Set if a misaligned phrase address is supplied (global address [2:0] != 000) |
| | FPVIOL | Set if the selected P-Flash sector is protected |
| | MGSTAT1 | Set if any errors have been encountered during the verify operation |
| | MGSTAT0 | Set if any non-correctable errors have been encountered during the verify operation |
| FERSTAT | EPVIOLIF | None |

### 27.4.2.11 Unsecure Flash Command

The Unsecure Flash command will erase the entire P-Flash and D-Flash memory space and, if the erase is successful, will release security.

**Table 27-53. Unsecure Flash Command FCCOB Requirements**

| CCOBIX[2:0] | FCCOB Parameters | |
|---|---|---|
| 000 | 0x0B | Not required |

Upon clearing CCIF to launch the Unsecure Flash command, the Memory Controller will erase the entire P-Flash and D-Flash memory space and verify that it is erased. If the Memory Controller verifies that the entire Flash memory space was properly erased, security will be released. If the erase verify is not successful, the Unsecure Flash operation sets MGSTAT1 and terminates without changing the security state. During the execution of this command (CCIF=0) the user must not write to any Flash module register. The CCIF flag is set after the Unsecure Flash operation has completed.

**Table 27-54. Unsecure Flash Command Error Handling**

| Register | Error Bit | Error Condition |
|---|---|---|
| FSTAT | ACCERR | Set if CCOBIX[2:0] != 000 at command launch |
| | | Set if a Load Data Field command sequence is currently active |
| | | Set if command not available in current mode (see Table 27-30) |
| | FPVIOL | Set if any area of the P-Flash memory is protected |
| | MGSTAT1 | Set if any errors have been encountered during the verify operation |
| | MGSTAT0 | Set if any non-correctable errors have been encountered during the verify operation |
| FERSTAT | EPVIOLIF | Set if any area of the buffer RAM EEE partition is protected |

### 27.4.2.12 Verify Backdoor Access Key Command

The Verify Backdoor Access Key command will only execute if it is enabled by the KEYEN bits in the FSEC register (see Table 27-11). The Verify Backdoor Access Key command releases security if user-supplied keys match those stored in the Flash security bytes of the Flash configuration field (see Table 27-3). The Verify Backdoor Access Key command must not be executed from the Flash block containing the backdoor comparison key to avoid code runaway.

**Table 27-55. Verify Backdoor Access Key Command FCCOB Requirements**

| CCOBIX[2:0] | FCCOB Parameters | |
|---|---|---|
| 000 | 0x0C | Not required |
| 001 | Key 0 | |
| 010 | Key 1 | |
| 011 | Key 2 | |
| 100 | Key 3 | |

Upon clearing CCIF to launch the Verify Backdoor Access Key command, the Memory Controller will check the FSEC KEYEN bits to verify that this command is enabled. If not enabled, the Memory Controller sets the ACCERR bit in the FSTAT register and terminates. If the command is enabled, the Memory Controller compares the key provided in FCCOB to the backdoor comparison key in the Flash configuration field with Key 0 compared to 0x7F_FF00, etc. If the backdoor keys match, security will be released. If the backdoor keys do not match, security is not released and all future attempts to execute the Verify Backdoor Access Key command are aborted (set ACCERR) until a reset occurs. The CCIF flag is set after the Verify Backdoor Access Key operation has completed.

**Table 27-56. Verify Backdoor Access Key Command Error Handling**

| Register | Error Bit | Error Condition |
|---|---|---|
| FSTAT | ACCERR | Set if CCOBIX[2:0] != 100 at command launch |
| | | Set if a Load Data Field command sequence is currently active |
| | | Set if an incorrect backdoor key is supplied |
| | | Set if backdoor key access has not been enabled (KEYEN[1:0] != 10, see Section 27.3.2.2) |
| | | Set if the backdoor key has mismatched since the last reset |
| | FPVIOL | None |
| | MGSTAT1 | None |
| | MGSTAT0 | None |
| FERSTAT | EPVIOLIF | None |

### 27.4.2.13 Set User Margin Level Command

The Set User Margin Level command causes the Memory Controller to set the margin level for future read operations of a specific P-Flash or D-Flash block.

**Table 27-57. Set User Margin Level Command FCCOB Requirements**

| CCOBIX[2:0] | FCCOB Parameters | |
|---|---|---|
| 000 | 0x0D | Global address [22:16] to identify the Flash block |
| 001 | Margin level setting | |

Upon clearing CCIF to launch the Set User Margin Level command, the Memory Controller will set the user margin level for the targeted block and then set the CCIF flag.

Valid margin level settings for the Set User Margin Level command are defined in Table 27-58.

**Table 27-58. Valid Set User Margin Level Settings**

| CCOB (CCOBIX=001) | Level Description |
|---|---|
| 0x0000 | Return to Normal Level |
| 0x0001 | User Margin-1 Level(1) |

**Table 27-58. Valid Set User Margin Level Settings**

| CCOB (CCOBIX=001) | Level Description |
|---|---|
| 0x0002 | User Margin-0 Level(2) |

1. Read margin to the erased state
2. Read margin to the programmed state

**Table 27-59. Set User Margin Level Command Error Handling**

| Register | Error Bit | Error Condition |
|---|---|---|
| FSTAT | ACCERR | Set if CCOBIX[2:0] != 001 at command launch |
| | | Set if a Load Data Field command sequence is currently active |
| | | Set if command not available in current mode (see Table 27-30) |
| | | Set if an invalid global address [22:16] is supplied |
| | | Set if an invalid margin level setting is supplied |
| | FPVIOL | None |
| | MGSTAT1 | None |
| | MGSTAT0 | None |
| FERSTAT | EPVIOLIF | None |

**NOTE**

User margin levels can be used to check that Flash memory contents have adequate margin for normal level read operations. If unexpected results are encountered when checking Flash memory contents at user margin levels, a potential loss of information has been detected.

### 27.4.2.14 Set Field Margin Level Command

The Set Field Margin Level command, valid in special modes only, causes the Memory Controller to set the margin level specified for future read operations of a specific P-Flash or D-Flash block.

**Table 27-60. Set Field Margin Level Command FCCOB Requirements**

| CCOBIX[2:0] | FCCOB Parameters | |
|---|---|---|
| 000 | 0x0E | Global address [22:16] to identify the Flash block |
| 001 | Margin level setting | |

Upon clearing CCIF to launch the Set Field Margin Level command, the Memory Controller will set the field margin level for the targeted block and then set the CCIF flag.

Valid margin level settings for the Set Field Margin Level command are defined in Table 27-61.

**Table 27-61. Valid Set Field Margin Level Settings**

| CCOB (CCOBIX=001) | Level Description |
|---|---|
| 0x0000 | Return to Normal Level |
| 0x0001 | User Margin-1 Level(1) |
| 0x0002 | User Margin-0 Level(2) |
| 0x0003 | Field Margin-1 Level[1] |
| 0x0004 | Field Margin-0 Level[2] |

1. Read margin to the erased state
2. Read margin to the programmed state

**Table 27-62. Set Field Margin Level Command Error Handling**

| Register | Error Bit | Error Condition |
|---|---|---|
| FSTAT | ACCERR | Set if CCOBIX[2:0] != 001 at command launch |
| | | Set if a Load Data Field command sequence is currently active |
| | | Set if command not available in current mode (see Table 27-30) |
| | | Set if an invalid global address [22:16] is supplied |
| | | Set if an invalid margin level setting is supplied |
| | FPVIOL | None |
| | MGSTAT1 | None |
| | MGSTAT0 | None |
| FERSTAT | EPVIOLIF | None |

**CAUTION**

Field margin levels must only be used during verify of the initial factory programming.

**NOTE**

Field margin levels can be used to check that Flash memory contents have adequate margin for data retention at the normal level setting. If unexpected results are encountered when checking Flash memory contents at field margin levels, the Flash memory contents should be erased and reprogrammed.

### 27.4.2.15 Full Partition D-Flash Command

The Full Partition D-Flash command allows the user to allocate sectors within the D-Flash block for applications and a partition within the buffer RAM for EEPROM access. The D-Flash block consists of 128 sectors with 256 bytes per sector.

**Table 27-63. Full Partition D-Flash Command FCCOB Requirements**

| CCOBIX[2:0] | FCCOB Parameters | |
|---|---|---|
| 000 | 0x0F | Not required |
| 001 | Number of 256 byte sectors for the D-Flash user partition (DFPART) | |
| 010 | Number of 256 byte sectors for buffer RAM EEE partition (ERPART) | |

Upon clearing CCIF to launch the Full Partition D-Flash command, the following actions are taken to define a partition within the D-Flash block for direct access (DFPART) and a partition within the buffer RAM for EEE use (ERPART):

- Validate the DFPART and ERPART values provided:
  - DFPART <= 128 (maximum number of 256 byte sectors in D-Flash block)
  - ERPART <= 16 (maximum number of 256 byte sectors in buffer RAM)
  - If ERPART > 0, 128 - DFPART >= 12 (minimum number of 256 byte sectors in the D-Flash block required to support EEE)
  - If ERPART > 0, ((128-DFPART)/ERPART) >= 8 (minimum ratio of D-Flash EEE space to buffer RAM EEE space to support EEE)
- Erase the D-Flash block and the EEE nonvolatile information register
- Program DFPART to the EEE nonvolatile information register at global address 0x12_0000 (see Table 27-7)
- Program a duplicate DFPART to the EEE nonvolatile information register at global address 0x12_0002 (see Table 27-7)
- Program ERPART to the EEE nonvolatile information register at global address 0x12_0004 (see Table 27-7)
- Program a duplicate ERPART to the EEE nonvolatile information register at global address 0x12_0006 (see Table 27-7)

The D-Flash user partition will start at global address 0x10_0000. The buffer RAM EEE partition will end at global address 0x13_FFFF. After the Full Partition D-Flash operation has completed, the CCIF flag will set.

Running the Full Partition D-Flash command a second time will result in the previous partition values and the entire D-Flash memory being erased. The data value written corresponds to the number of 256 byte sectors allocated for either direct D-Flash access (DFPART) or buffer RAM EEE access (ERPART).

**Table 27-64. Full Partition D-Flash Command Error Handling**

| Register | Error Bit | Error Condition |
|---|---|---|
| FSTAT | ACCERR | Set if CCOBIX[2:0] != 010 at command launch |
| | | Set if a Load Data Field command sequence is currently active |
| | | Set if command not available in current mode (see Table 27-30) |
| | | Set if an invalid DFPART or ERPART selection is supplied |
| | FPVIOL | None |
| | MGSTAT1 | Set if any errors have been encountered during the read |
| | MGSTAT0 | Set if any non-correctable errors have been encountered during the read |
| FERSTAT | EPVIOLIF | None |

### 27.4.2.16 Erase Verify D-Flash Section Command

The Erase Verify D-Flash Section command will verify that a section of code in the D-Flash user partition is erased. The Erase Verify D-Flash Section command defines the starting point of the data to be verified and the number of words.

**Table 27-65. Erase Verify D-Flash Section Command FCCOB Requirements**

| CCOBIX[2:0] | FCCOB Parameters | |
|---|---|---|
| 000 | 0x10 | Global address [22:16] to identify the D-Flash block |
| 001 | Global address [15:0] of the first word to be verified | |
| 010 | Number of words to be verified | |

Upon clearing CCIF to launch the Erase Verify D-Flash Section command, the Memory Controller will verify the selected section of D-Flash memory is erased. The CCIF flag will set after the Erase Verify D-Flash Section operation has completed.

**Table 27-66. Erase Verify D-Flash Section Command Error Handling**

| Register | Error Bit | Error Condition |
|---|---|---|
| FSTAT | ACCERR | Set if CCOBIX[2:0] != 010 at command launch |
| | | Set if a Load Data Field command sequence is currently active |
| | | Set if command not available in current mode (see Table 27-30) |
| | | Set if an invalid global address [22:0] is supplied |
| | | Set if a misaligned word address is supplied (global address [0] != 0) |
| | | Set if the global address [22:0] points to an area of the D-Flash EEE partition |
| | | Set if the requested section breaches the end of the D-Flash block or goes into the D-Flash EEE partition |
| | FPVIOL | None |
| | MGSTAT1 | Set if any errors have been encountered during the read |
| | MGSTAT0 | Set if any non-correctable errors have been encountered during the read |
| FERSTAT | EPVIOLIF | None |

### 27.4.2.17 Program D-Flash Command

The Program D-Flash operation programs one to four previously erased words in the D-Flash user partition. The Program D-Flash operation will confirm that the targeted location(s) were successfully programmed upon completion.

**CAUTION**

A Flash word must be in the erased state before being programmed. Cumulative programming of bits within a Flash word is not allowed.

**Table 27-67. Program D-Flash Command FCCOB Requirements**

| CCOBIX[2:0] | FCCOB Parameters | |
|---|---|---|
| 000 | 0x11 | Global address [22:16] to identify the D-Flash block |
| 001 | Global address [15:0] of word to be programmed | |
| 010 | Word 0 program value | |
| 011 | Word 1 program value, if desired | |
| 100 | Word 2 program value, if desired | |
| 101 | Word 3 program value, if desired | |

Upon clearing CCIF to launch the Program D-Flash command, the user-supplied words will be transferred to the Memory Controller and be programmed. The CCOBIX index value at Program D-Flash command launch determines how many words will be programmed in the D-Flash block. No protection checks are made in the Program D-Flash operation on the D-Flash block, only access error checks. The CCIF flag is set when the operation has completed.

**Table 27-68. Program D-Flash Command Error Handling**

| Register | Error Bit | Error Condition |
|---|---|---|
| FSTAT | ACCERR | Set if CCOBIX[2:0] < 010 at command launch |
| | | Set if CCOBIX[2:0] > 101 at command launch |
| | | Set if a Load Data Field command sequence is currently active |
| | | Set if command not available in current mode (see Table 27-30) |
| | | Set if an invalid global address [22:0] is supplied |
| | | Set if a misaligned word address is supplied (global address [0] != 0) |
| | | Set if the global address [22:0] points to an area in the D-Flash EEE partition |
| | | Set if the requested group of words breaches the end of the D-Flash block or goes into the D-Flash EEE partition |
| | FPVIOL | None |
| | MGSTAT1 | Set if any errors have been encountered during the verify operation |
| | MGSTAT0 | Set if any non-correctable errors have been encountered during the verify operation |
| FERSTAT | EPVIOLIF | None |

### 27.4.2.18 Erase D-Flash Sector Command

The Erase D-Flash Sector operation will erase all addresses in a sector of the D-Flash user partition.

**Table 27-69. Erase D-Flash Sector Command FCCOB Requirements**

| CCOBIX[2:0] | FCCOB Parameters | |
|---|---|---|
| 000 | 0x12 | Global address [22:16] to identify D-Flash block |
| 001 | Global address [15:0] anywhere within the sector to be erased. See Section 27.1.2.2 for D-Flash sector size. | |

Upon clearing CCIF to launch the Erase D-Flash Sector command, the Memory Controller will erase the selected Flash sector and verify that it is erased. The CCIF flag will set after the Erase D-Flash Sector operation has completed.

**Table 27-70. Erase D-Flash Sector Command Error Handling**

| Register | Error Bit | Error Condition |
|---|---|---|
| FSTAT | ACCERR | Set if CCOBIX[2:0] != 001 at command launch |
| | | Set if a Load Data Field command sequence is currently active |
| | | Set if command not available in current mode (see Table 27-30) |
| | | Set if an invalid global address [22:0] is supplied |
| | | Set if a misaligned word address is supplied (global address [0] != 0) |
| | | Set if the global address [22:0] points to the D-Flash EEE partition |
| | FPVIOL | None |
| | MGSTAT1 | Set if any errors have been encountered during the verify operation |
| | MGSTAT0 | Set if any non-correctable errors have been encountered during the verify operation |
| FERSTAT | EPVIOLIF | None |

### 27.4.2.19 Enable EEPROM Emulation Command

The Enable EEPROM Emulation command causes the Memory Controller to enable EEE activity. EEE activity is disabled after any reset.

**Table 27-71. Enable EEPROM Emulation Command FCCOB Requirements**

| CCOBIX[2:0] | FCCOB Parameters | |
|---|---|---|
| 000 | 0x13 | Not required |

Upon clearing CCIF to launch the Enable EEPROM Emulation command, the CCIF flag will set after the Memory Controller enables EEE operations using the contents of the EEE tag RAM and tag counter. The Full Partition D-Flash or the Partition D-Flash command must be run prior to launching the Enable EEPROM Emulation command.

**Table 27-72. Enable EEPROM Emulation Command Error Handling**

| Register | Error Bit | Error Condition |
|---|---|---|
| FSTAT | ACCERR | Set if CCOBIX[2:0] != 000 at command launch |
| | | Set if a Load Data Field command sequence is currently active |
| | | Set if Full Partition D-Flash or Partition D-Flash command not previously run |
| | FPVIOL | None |
| | MGSTAT1 | None |
| | MGSTAT0 | None |
| FERSTAT | EPVIOLIF | None |

### 27.4.2.20 Disable EEPROM Emulation Command

The Disable EEPROM Emulation command causes the Memory Controller to suspend current EEE activity.

**Table 27-73. Disable EEPROM Emulation Command FCCOB Requirements**

| CCOBIX[2:0] | FCCOB Parameters | |
|---|---|---|
| 000 | 0x14 | Not required |

Upon clearing CCIF to launch the Disable EEPROM Emulation command, the Memory Controller will halt EEE operations at the next convenient point without clearing the EEE tag RAM or tag counter before setting the CCIF flag.

**Table 27-74. Disable EEPROM Emulation Command Error Handling**

| Register | Error Bit | Error Condition |
|---|---|---|
| FSTAT | ACCERR | Set if CCOBIX[2:0] != 000 at command launch |
| | | Set if a Load Data Field command sequence is currently active |
| | | Set if Full Partition D-Flash or Partition D-Flash command not previously run |
| | FPVIOL | None |
| | MGSTAT1 | None |
| | MGSTAT0 | None |
| FERSTAT | EPVIOLIF | None |

### 27.4.2.21 EEPROM Emulation Query Command

The EEPROM Emulation Query command returns EEE partition and status variables.

**Table 27-75. EEPROM Emulation Query Command FCCOB Requirements**

| CCOBIX[2:0] | FCCOB Parameters | |
|---|---|---|
| 000 | 0x15 | Not required |
| 001 | Return DFPART | |
| 010 | Return ERPART | |
| 011 | Return ECOUNT(1) | |
| 100 | Return Dead Sector Count | Return Ready Sector Count |

1. Indicates sector erase count

Upon clearing CCIF to launch the EEPROM Emulation Query command, the CCIF flag will set after the EEE partition and status variables are stored in the FCCOBIX register.If the Emulation Query command is executed prior to partitioning (Partition D-Flash Command Section 27.4.2.15), the following reset values are returned: DFPART = 0x_FFFF, ERPART = 0x_FFFF, ECOUNT = 0x_FFFF, Dead Sector Count = 0x_00, Ready Sector Count = 0x_00.

**Table 27-76. EEPROM Emulation Query Command Error Handling**

| Register | Error Bit | Error Condition |
|---|---|---|
| FSTAT | ACCERR | Set if CCOBIX[2:0] != 000 at command launch |
| | | Set if a Load Data Field command sequence is currently active |
| | | Set if command not available in current mode (see Table 27-30) |
| | FPVIOL | None |
| | MGSTAT1 | None |
| | MGSTAT0 | None |
| FERSTAT | EPVIOLIF | None |

### 27.4.2.22 Partition D-Flash Command

The Partition D-Flash command allows the user to allocate sectors within the D-Flash block for applications and a partition within the buffer RAM for EEPROM access. The D-Flash block consists of 128 sectors with 256 bytes per sector. The Erase All Blocks command must be run prior to launching the Partition D-Flash command.

**Table 27-77. Partition D-Flash Command FCCOB Requirements**

| CCOBIX[2:0] | FCCOB Parameters | |
|---|---|---|
| 000 | 0x20 | Not required |
| 001 | Number of 256 byte sectors for the D-Flash user partition (DFPART) | |
| 010 | Number of 256 byte sectors for buffer RAM EEE partition (ERPART) | |

Upon clearing CCIF to launch the Partition D-Flash command, the following actions are taken to define a partition within the D-Flash block for direct access (DFPART) and a partition within the buffer RAM for EEE use (ERPART):

- Validate the DFPART and ERPART values provided:
  - DFPART <= 128 (maximum number of 256 byte sectors in D-Flash block)
  - ERPART <= 16 (maximum number of 256 byte sectors in buffer RAM)
  - If ERPART > 0, 128 - DFPART >= 12 (minimum number of 256 byte sectors in the D-Flash block required to support EEE)
  - If ERPART > 0, ((128-DFPART)/ERPART) >= 8 (minimum ratio of D-Flash EEE space to buffer RAM EEE space to support EEE)
- Erase verify the D-Flash block and the EEE nonvolatile information register
- Program DFPART to the EEE nonvolatile information register at global address 0x12_0000 (see Table 27-7)

- Program a duplicate DFPART to the EEE nonvolatile information register at global address 0x12_0002 (see Table 27-7)
- Program ERPART to the EEE nonvolatile information register at global address 0x12_0004 (see Table 27-7)
- Program a duplicate ERPART to the EEE nonvolatile information register at global address 0x12_0006 (see Table 27-7)

The D-Flash user partition will start at global address 0x10_0000. The buffer RAM EEE partition will end at global address 0x13_FFFF. After the Partition D-Flash operation has completed, the CCIF flag will set.

Running the Partition D-Flash command a second time will result in the ACCERR bit within the FSTAT register being set. The data value written corresponds to the number of 256 byte sectors allocated for either direct D-Flash access (DFPART) or buffer RAM EEE access (ERPART).

**Table 27-78. Partition D-Flash Command Error Handling**

| Register | Error Bit | Error Condition |
|---|---|---|
| FSTAT | ACCERR | Set if CCOBIX[2:0] != 010 at command launch |
| | | Set if a Load Data Field command sequence is currently active |
| | | Set if command not available in current mode (see Table 27-30) |
| | | Set if partitions have already been defined |
| | | Set if an invalid DFPART or ERPART selection is supplied |
| | FPVIOL | None |
| | MGSTAT1 | Set if any errors have been encountered during the read |
| | MGSTAT0 | Set if any non-correctable errors have been encountered during the read |
| FERSTAT | EPVIOLIF | None |

## 27.4.3 Interrupts

The Flash module can generate an interrupt when a Flash command operation has completed or when a Flash command operation has detected an EEE error or an ECC fault.

**Table 27-79. Flash Interrupt Sources**

| Interrupt Source | Interrupt Flag | Local Enable | Global (CCR) Mask |
|---|---|---|---|
| Flash Command Complete | CCIF (FSTAT register) | CCIE (FCNFG register) | I Bit |
| Flash EEE Erase Error | ERSERIF (FERSTAT register) | ERSERIE (FERCNFG register) | I Bit |
| Flash EEE Program Error | PGMERIF (FERSTAT register) | PGMERIE (FERCNFG register) | I Bit |
| Flash EEE Protection Violation | EPVIOLIF (FERSTAT register) | EPVIOLIE (FERCNFG register) | I Bit |
| Flash EEE Error Type 1 Violation | ERSVIF1 (FERSTAT register) | ERSVIE1 (FERCNFG register) | I Bit |
| Flash EEE Error Type 0 Violation | ERSVIF0 (FERSTAT register) | ERSVIE0 (FERCNFG register) | I Bit |
| ECC Double Bit Fault on Flash Read | DFDIF (FERSTAT register) | DFDIE (FERCNFG register) | I Bit |
| ECC Single Bit Fault on Flash Read | SFDIF (FERSTAT register) | SFDIE (FERCNFG register) | I Bit |

**NOTE**

Vector addresses and their relative interrupt priority are determined at the MCU level.

### 27.4.3.1 Description of Flash Interrupt Operation

The Flash module uses the CCIF flag in combination with the CCIE interrupt enable bit to generate the Flash command interrupt request. The Flash module uses the ERSEIF, PGMEIF, EPVIOLIF, ERSVIF1, ERSVIF0, DFDIF and SFDIF flags in combination with the ERSEIE, PGMEIE, EPVIOLIE, ERSVIE1, ERSVIE0, DFDIE and SFDIE interrupt enable bits to generate the Flash error interrupt request. For a detailed description of the register bits involved, refer to Section 27.3.2.5, “Flash Configuration Register (FCNFG)”, Section 27.3.2.6, “Flash Error Configuration Register (FERCNFG)”, Section 27.3.2.7, “Flash Status Register (FSTAT)”, and Section 27.3.2.8, “Flash Error Status Register (FERSTAT)”.

The logic used for generating the Flash module interrupts is shown in Figure 27-27.

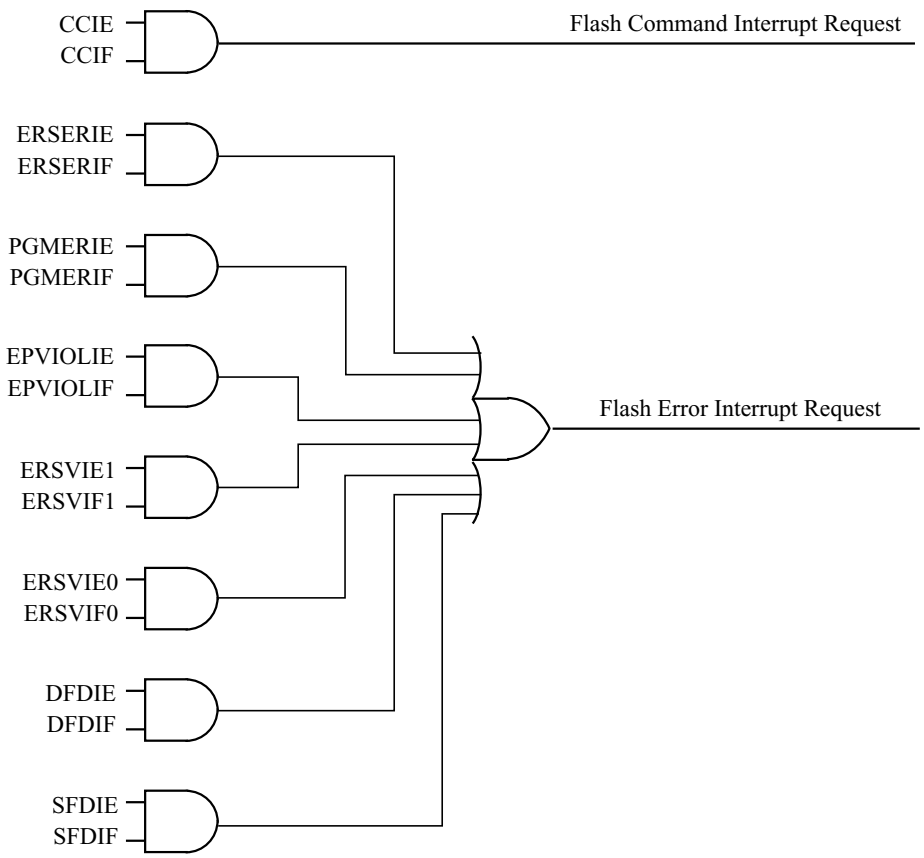



**Figure 27-27. Flash Module Interrupts Implementation**

### 27.4.4 Wait Mode

The Flash module is not affected if the MCU enters wait mode. The Flash module can recover the MCU from wait via the CCIF interrupt (see Section 27.4.3, “Interrupts”).

### 27.4.5 Stop Mode

If a Flash command is active (CCIF = 0) or an EE-Emulation operation is pending when the MCU requests stop mode, the current Flash operation will be completed before the CPU is allowed to enter stop mode.

## 27.5 Security

The Flash module provides security information to the MCU. The Flash security state is defined by the SEC bits of the FSEC register (see Table 27-12). During reset, the Flash module initializes the FSEC register using data read from the security byte of the Flash configuration field at global address 0x7F_FF0F.

The security state out of reset can be permanently changed by programming the security byte of the Flash configuration field. This assumes that you are starting from a mode where the necessary P-Flash erase and program commands are available and that the upper region of the P-Flash is unprotected. If the Flash security byte is successfully programmed, its new value will take affect after the next MCU reset.

The following subsections describe these security-related subjects:

- Unsecuring the MCU using Backdoor Key Access
- Unsecuring the MCU in Special Single Chip Mode using BDM
- Mode and Security Effects on Flash Command Availability

### 27.5.1 Unsecuring the MCU using Backdoor Key Access

The MCU may be unsecured by using the backdoor key access feature which requires knowledge of the contents of the backdoor keys (four 16-bit words programmed at addresses 0x7F_FF00–0x7F_FF07). If the KEYEN[1:0] bits are in the enabled state (see Section 27.3.2.2), the Verify Backdoor Access Key command (see Section 27.4.2.12) allows the user to present four prospective keys for comparison to the keys stored in the Flash memory via the Memory Controller. If the keys presented in the Verify Backdoor Access Key command match the backdoor keys stored in the Flash memory, the SEC bits in the FSEC register (see Table 27-12) will be changed to unsecure the MCU. Key values of 0x0000 and 0xFFFF are not permitted as backdoor keys. While the Verify Backdoor Access Key command is active, P-Flash block 0 will not be available for read access and will return invalid data.

The user code stored in the P-Flash memory must have a method of receiving the backdoor keys from an external stimulus. This external stimulus would typically be through one of the on-chip serial ports.

If the KEYEN[1:0] bits are in the enabled state (see Section 27.3.2.2), the MCU can be unsecured by the backdoor key access sequence described below:

1. Follow the command sequence for the Verify Backdoor Access Key command as explained in Section 27.4.2.12
2. If the Verify Backdoor Access Key command is successful, the MCU is unsecured and the SEC[1:0] bits in the FSEC register are forced to the unsecure state of 10

The Verify Backdoor Access Key command is monitored by the Memory Controller and an illegal key will prohibit future use of the Verify Backdoor Access Key command. A reset of the MCU is the only method to re-enable the Verify Backdoor Access Key command.

After the backdoor keys have been correctly matched, the MCU will be unsecured. After the MCU is unsecured, the sector containing the Flash security byte can be erased and the Flash security byte can be reprogrammed to the unsecure state, if desired.

In the unsecure state, the user has full control of the contents of the backdoor keys by programming addresses 0x7F_FF00–0x7F_FF07 in the Flash configuration field.

The security as defined in the Flash security byte (0x7F_FF0F) is not changed by using the Verify Backdoor Access Key command sequence. The backdoor keys stored in addresses 0x7F_FF00–0x7F_FF07 are unaffected by the Verify Backdoor Access Key command sequence. After the next reset of the MCU, the security state of the Flash module is determined by the Flash security byte

(0x7F_FF0F). The Verify Backdoor Access Key command sequence has no effect on the program and erase protections defined in the Flash protection register, FPROT.

### 27.5.2 Unsecuring the MCU in Special Single Chip Mode using BDM

The MCU can be unsecured in special single chip mode by erasing the P-Flash and D-Flash memory by one of the following methods:

- Reset the MCU into special single chip mode, delay while the erase test is performed by the BDM, send BDM commands to disable protection in the P-Flash and D-Flash memory, and execute the Erase All Blocks command write sequence to erase the P-Flash and D-Flash memory.
- Reset the MCU into special expanded wide mode, disable protection in the P-Flash and D-Flash memory and run code from external memory to execute the Erase All Blocks command write sequence to erase the P-Flash and D-Flash memory.

After the CCIF flag sets to indicate that the Erase All Blocks operation has completed, reset the MCU into special single chip mode. The BDM will execute the Erase Verify All Blocks command write sequence to verify that the P-Flash and D-Flash memory is erased. If the P-Flash and D-Flash memory are verified as erased the MCU will be unsecured. All BDM commands will be enabled and the Flash security byte may be programmed to the unsecure state by the following method:

- Send BDM commands to execute a ‘Program P-Flash’ command sequence to program the Flash security byte to the unsecured state and reset the MCU.

### 27.5.3 Mode and Security Effects on Flash Command Availability

The availability of Flash module commands depends on the MCU operating mode and security state as shown in Table 27-30.

## 27.6 Initialization

On each system reset the Flash module executes a reset sequence which establishes initial values for the Flash Block Configuration Parameters, the FPROT and DFPROT protection registers, and the FOPT and FSEC registers. The Flash module reverts to built-in default values that leave the module in a fully protected and secured state if errors are encountered during execution of the reset sequence. If a double bit fault is detected during the reset sequence, both MGSTAT bits in the FSTAT register will be set. The ACCERR bit in the FSTAT register is set if errors are encountered while initializing the EEE buffer ram during the reset sequence.

CCIF remains clear throughout the reset sequence. The Flash module holds off all CPU access for the initial portion of the reset sequence. While Flash reads are possible when the hold is removed, writes to the FCCOBIX, FCCOBHI, and FCCOBLO registers are ignored to prevent command activity while the Memory Controller remains busy. Completion of the reset sequence is marked by setting CCIF high which enables writes to the FCCOBIX, FCCOBHI, and FCCOBLO registers to launch any available Flash command.

If a reset occurs while any Flash command is in progress, that command will be immediately aborted. The state of the word being programmed or the sector/block being erased is not guaranteed.

# Chapter 28
# 768 KByte Flash Module (S12XFTM768K4V2)

**Table 28-1. Revision History**

| Revision Number | Revision Date | Sections Affected | Description of Changes |
|---|---|---|---|
| V02.09 | 29 Nov 2007 | | - Cleanup |
| V02.10 | 19 Dec 2007 | 28.4.2/28-1113<br><br>28.4.2/28-1113<br><br>28.3.1/28-1082 | - Updated Command Error Handling tables based on parent-child relationship with FTM1024K5<br>- Corrected Error Handling table for Full Partition D-Flash, Partition D-Flash, and EEPROM Emulation Query commands<br>- Corrected P-Flash Memory Addressing table |
| V02.11 | 25 Sep 2009 | 28.1/28-1077<br>28.3.2.1/28-1089<br>28.4.2.4/28-1116<br><br>28.4.2.7/28-1119<br><br>28.4.2.12/28-1123<br><br>28.4.2.12/28-1123<br>28.4.2.12/28-1123<br>28.4.2.20/28-1132<br><br><br>28.3.2/28-1087<br>28.3.2.1/28-1089<br>28.4.1.2/28-1108<br>28.6/28-1138 | - Clarify single bit fault correction for P-Flash phrase<br>- Expand FDIV vs OSCCLK Frequency table<br>- Add statement concerning code runaway when executing Read Once command from Flash block containing associated fields<br>- Add statement concerning code runaway when executing Program Once command from Flash block containing associated fields<br>- Add statement concerning code runaway when executing Verify Backdoor Access Key command from Flash block containing associated fields<br>- Relate Key 0 to associated Backdoor Comparison Key address<br>- Change “power down reset” to “reset”<br>- Add ACCERR condition for Disable EEPROM Emulation command<br>The following changes were made to clarify module behavior related to Flash register access during reset sequence and while Flash commands are active:<br>- Add caution concerning register writes while command is active<br>- Writes to FCLKDIV are allowed during reset sequence while CCIF is clear<br>- Add caution concerning register writes while command is active<br>- Writes to FCCOBIX, FCCOBHI, FCCOBLO registers are ignored during reset sequence |

## 28.1 Introduction

The FTM768K4 module implements the following:

- 768 Kbytes of P-Flash (Program Flash) memory, consisting of 4 physical Flash blocks, intended primarily for nonvolatile code storage

- 32 Kbytes of D-Flash (Data Flash) memory, consisting of 1 physical Flash block, that can be used as nonvolatile storage to support the built-in hardware scheme for emulated EEPROM, as basic Flash memory primarily intended for nonvolatile data storage, or as a combination of both
- 4 Kbytes of buffer RAM, consisting of 1 physical RAM block, that can be used as emulated EEPROM using a built-in hardware scheme, as basic RAM, or as a combination of both

The Flash memory is ideal for single-supply applications allowing for field reprogramming without requiring external high voltage sources for program or erase operations. The Flash module includes a memory controller that executes commands to modify Flash memory contents or configure module resources for emulated EEPROM operation. The user interface to the memory controller consists of the indexed Flash Common Command Object (FCCOB) register which is written to with the command, global address, data, and any required command parameters. The memory controller must complete the execution of a command before the FCCOB register can be written to with a new command.

**CAUTION**

A Flash word or phrase must be in the erased state before being programmed. Cumulative programming of bits within a Flash word or phrase is not allowed.

The RAM and Flash memory may be read as bytes, aligned words, or misaligned words. Read access time is one bus cycle for bytes and aligned words, and two bus cycles for misaligned words. For Flash memory, an erased bit reads 1 and a programmed bit reads 0.

It is not possible to read from a Flash block while any command is executing on that specific Flash block. It is possible to read from a Flash block while a command is executing on a different Flash block.

Both P-Flash and D-Flash memories are implemented with Error Correction Codes (ECC) that can resolve single bit faults and detect double bit faults. For P-Flash memory, the ECC implementation requires that programming be done on an aligned 8 byte basis (a Flash phrase). Since P-Flash memory is always read by phrase, only one single bit fault in the phrase containing the byte or word accessed will be corrected.

### 28.1.1 Glossary

**Buffer RAM** — The buffer RAM constitutes the volatile memory store required for EEE. Memory space in the buffer RAM not required for EEE can be partitioned to provide volatile memory space for applications.

**Command Write Sequence** — An MCU instruction sequence to execute built-in algorithms (including program and erase) on the Flash memory.

**D-Flash Memory** — The D-Flash memory constitutes the nonvolatile memory store required for EEE. Memory space in the D-Flash memory not required for EEE can be partitioned to provide nonvolatile memory space for applications.

**D-Flash Sector** — The D-Flash sector is the smallest portion of the D-Flash memory that can be erased. The D-Flash sector consists of four 64 byte rows for a total of 256 bytes.

**EEE (Emulated EEPROM)** — A method to emulate the small sector size features and endurance characteristics associated with an EEPROM.

**EEE IFR** — Nonvolatile information register located in the D-Flash block that contains data required to partition the D-Flash memory and buffer RAM for EEE. The EEE IFR is visible in the global memory map by setting the EEEIFRON bit in the MMCCTL1 register.

**NVM Command Mode** — An NVM mode using the CPU to setup the FCCOB register to pass parameters required for Flash command execution.

**Phrase** — An aligned group of four 16-bit words within the P-Flash memory. Each phrase includes eight ECC bits for single bit fault correction and double bit fault detection within the phrase.

**P-Flash Memory** — The P-Flash memory constitutes the main nonvolatile memory store for applications.

**P-Flash Sector** — The P-Flash sector is the smallest portion of the P-Flash memory that can be erased. Each P-Flash sector contains 1024 bytes.

**Program IFR** — Nonvolatile information register located in the P-Flash block that contains the Device ID, Version ID, and the Program Once field. The Program IFR is visible in the global memory map by setting the PGMIFRON bit in the MMCCTL1 register.

## 28.1.2 Features

### 28.1.2.1 P-Flash Features

- 768 Kbytes of P-Flash memory composed of two 256 Kbyte Flash blocks and two 128 Kbyte Flash blocks. The 256 Kbyte Flash block consists of two 128 Kbyte sections each divided into 128 sectors of 1024 bytes. The 128 Kbyte Flash blocks are each divided into 128 sectors of 1024 bytes.
- Single bit fault correction and double bit fault detection within a 64-bit phrase during read operations
- Automated program and erase algorithm with verify and generation of ECC parity bits
- Fast sector erase and phrase program operation
- Ability to program up to one phrase in each P-Flash block simultaneously
- Flexible protection scheme to prevent accidental program or erase of P-Flash memory

### 28.1.2.2 D-Flash Features

- Up to 32 Kbytes of D-Flash memory with 256 byte sectors for user access
- Dedicated commands to control access to the D-Flash memory over EEE operation
- Single bit fault correction and double bit fault detection within a word during read operations
- Automated program and erase algorithm with verify and generation of ECC parity bits
- Fast sector erase and word program operation

- Ability to program up to four words in a burst sequence

#### 28.1.2.3 Emulated EEPROM Features

- Up to 4 Kbytes of emulated EEPROM (EEE) accessible as 4 Kbytes of RAM
- Flexible protection scheme to prevent accidental program or erase of data
- Automatic EEE file handling using an internal Memory Controller
- Automatic transfer of valid EEE data from D-Flash memory to buffer RAM on reset
- Ability to monitor the number of outstanding EEE related buffer RAM words left to be programmed into D-Flash memory
- Ability to disable EEE operation and allow priority access to the D-Flash memory
- Ability to cancel all pending EEE operations and allow priority access to the D-Flash memory

#### 28.1.2.4 User Buffer RAM Features

- Up to 4 Kbytes of RAM for user access

#### 28.1.2.5 Other Flash Module Features

- No external high-voltage power supply required for Flash memory program and erase operations
- Interrupt generation on Flash command completion and Flash error detection
- Security mechanism to prevent unauthorized access to the Flash memory

### 28.1.3 Block Diagram

The block diagram of the Flash module is shown in Figure 28-1.

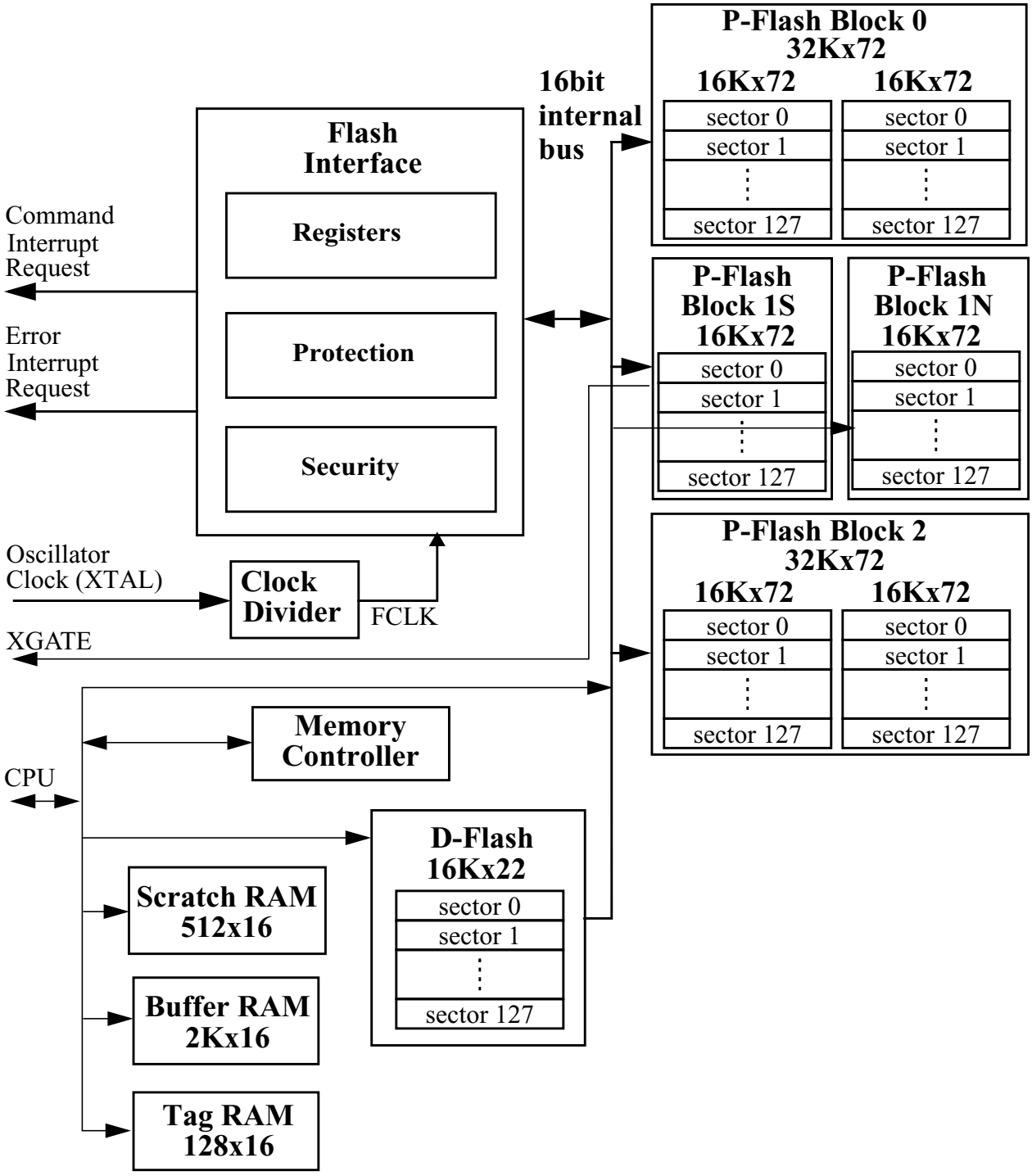



**Figure 28-1. FTM768K4 Block Diagram**

## 28.2 External Signal Description

The Flash module contains no signals that connect off-chip.

## 28.3 Memory Map and Registers

This section describes the memory map and registers for the Flash module. Read data from unimplemented memory space in the Flash module is undefined. Write access to unimplemented or reserved memory space in the Flash module will be ignored by the Flash module.

### 28.3.1 Module Memory Map

The S12X architecture places the P-Flash memory between global addresses 0x74_0000 and 0x7F_FFFF as shown in Table 28-2. The P-Flash memory map is shown in Figure 28-2.

**Table 28-2. P-Flash Memory Addressing**

| Global Address | Size (Bytes) | Description |
|---|---|---|
| 0x7C_0000 – 0x7F_FFFF | 256 K | P-Flash Block 0<br>Contains Flash Configuration Field (see Table 28-3) |
| 0x7A_0000 – 0x7B_FFFF | 128 K | P-Flash Block 1N |
| 0x78_0000 – 0x79_FFFF | 128 K | P-Flash Block 1S |
| 0x74_0000 – 0x77_FFFF | 256 K | P-Flash Block 2 |
| 0x70_0000 – 0x73_FFFF | 256 K | No P-Flash Memory |

The FPROT register, described in Section 28.3.2.9, can be set to protect regions in the Flash memory from accidental program or erase. Three separate memory regions, one growing upward from global address 0x7F_8000 in the Flash memory (called the lower region), one growing downward from global address 0x7F_FFFF in the Flash memory (called the higher region), and the remaining addresses in the Flash memory, can be activated for protection. The Flash memory addresses covered by these protectable regions are shown in the P-Flash memory map. The higher address region is mainly targeted to hold the boot loader code since it covers the vector space. Default protection settings as well as security information that allows the MCU to restrict access to the Flash module are stored in the Flash configuration field as described in Table 28-3.

**Table 28-3. Flash Configuration Field(1)**

| Global Address | Size (Bytes) | Description |
|---|---|---|
| 0x7F_FF00 – 0x7F_FF07 | 8 | Backdoor Comparison Key<br>Refer to Section 28.4.2.12, “Verify Backdoor Access Key Command,” and Section 28.5.1, “Unsecuring the MCU using Backdoor Key Access” |
| 0x7F_FF08 – 0x7F_FF0B(2) | 4 | Reserved |
| 0x7F_FF0C[2] | 1 | P-Flash Protection byte.<br>Refer to Section 28.3.2.9, “P-Flash Protection Register (FPROT)” |
| 0x7F_FF0D[2] | 1 | EEE Protection byte<br>Refer to Section 28.3.2.10, “EEE Protection Register (EPROT)” |

**Table 28-3. Flash Configuration Field[1]**

| Global Address | Size (Bytes) | Description |
|---|---|---|
| 0x7F_FF0E[2] | 1 | Flash Nonvolatile byte<br>Refer to Section 28.3.2.14, “Flash Option Register (FOPT)” |
| 0x7F_FF0F[2] | 1 | Flash Security byte<br>Refer to Section 28.3.2.2, “Flash Security Register (FSEC)” |

1. Older versions may have swapped protection byte addresses
2. 0x7FF08 - 0x7F_FF0F form a Flash phrase and must be programmed in a single command write sequence. Each byte in the 0x7F_FF08 - 0x7F_FF0B reserved field should be programmed to 0xFF.

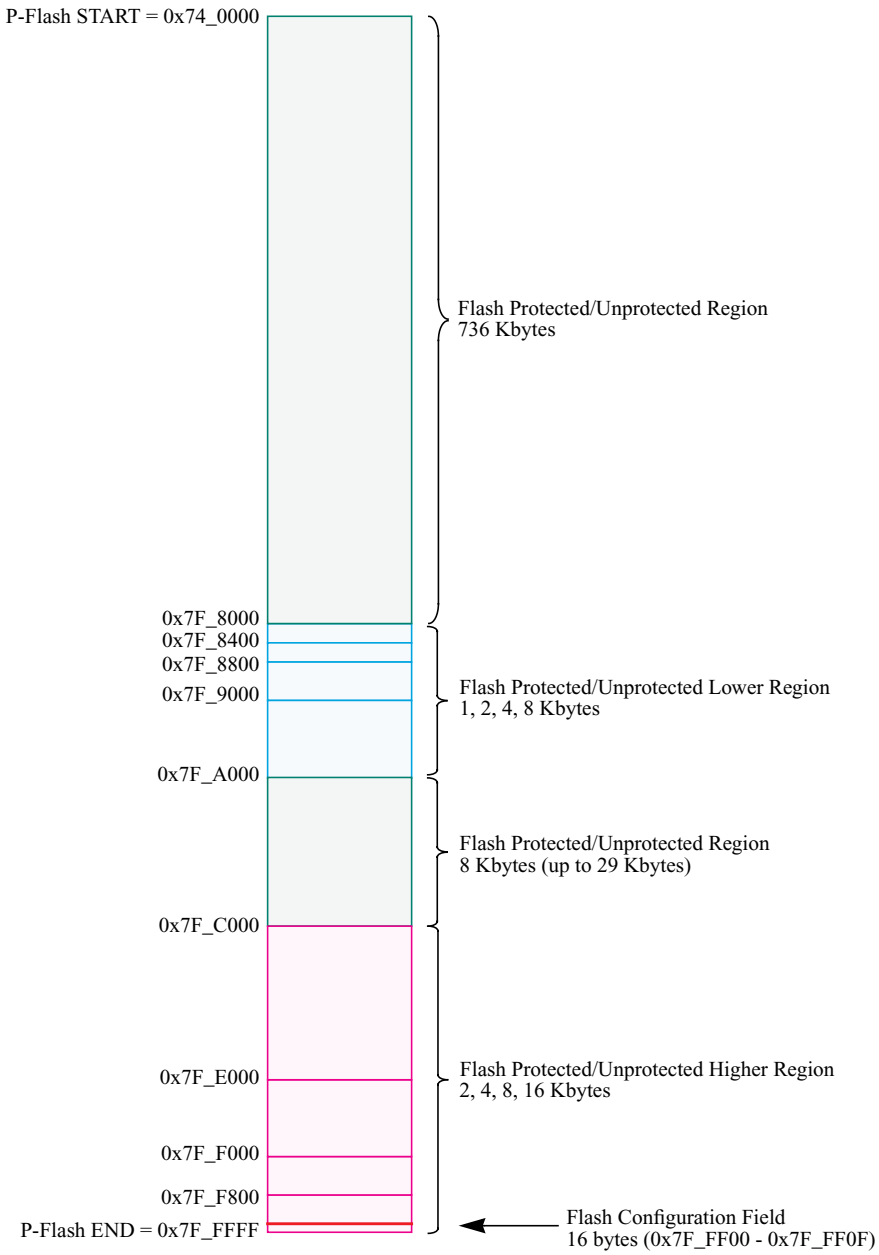



**Figure 28-2. P-Flash Memory Map**

**Table 28-4. Program IFR Fields**

| Global Address (PGMIFRON) | Size (Bytes) | Field Description |
|---|---|---|
| 0x40_0000 – 0x40_0007 | 8 | Device ID |
| 0x40_0008 – 0x40_00E7 | 224 | Reserved |
| 0x40_00E8 – 0x40_00E9 | 2 | Version ID |
| 0x40_00EA – 0x40_00FF | 22 | Reserved |
| 0x40_0100 – 0x40_013F | 64 | Program Once Field<br>Refer to Section 28.4.2.7, “Program Once Command” |
| 0x40_0140 – 0x40_01FF | 192 | Reserved |

**Table 28-5. P-Flash IFR Accessibility**

| Global Address (PGMIFRON) | Size (Bytes) | Accessed From |
|---|---|---|
| 0x40_0000 – 0x40_01FF | 512 | XBUS0 (PBLK0S)(1) |
| 0x40_0200 – 0x40_03FF | 512 | Unimplemented |
| 0x40_0400 – 0x40_05FF | 512 | XBUS0 (PBLK1N) |
| 0x40_0600 – 0x40_07FF | 512 | XBUS1 (PBLK1S) |
| 0x40_0800 – 0x40_09FF | 512 | XBUS0 (PBLK2S) |
| 0x40_0A00 – 0x40_0BFF | 512 | Unimplemented |

1. Refer to Table 28-4 for more details.

**Table 28-6. EEE Resource Fields**

| Global Address | Size (Bytes) | Description |
|---|---|---|
| 0x10_0000 – 0x10_7FFF | 32,768 | D-Flash Memory (User and EEE) |
| 0x10_8000 – 0x11_FFFF | 98,304 | Reserved |
| 0x12_0000 – 0x12_007F | 128 | EEE Nonvolatile Information Register (EEEIFRON(1) = 1) |
| 0x12_0080 – 0x12_0FFF | 3,968 | Reserved |
| 0x12_1000 – 0x12_1EFF | 3,840 | Reserved |
| 0x12_1F00 – 0x12_1FFF | 256 | EEE Tag RAM (TMGRAMON[1] = 1) |
| 0x12_2000 – 0x12_3BFF | 7,168 | Reserved |
| 0x12_3C00 – 0x12_3FFF | 1,024 | Memory Controller Scratch RAM (TMGRAMON[1] = 1) |
| 0x12_4000 – 0x12_DFFF | 40,960 | Reserved |
| 0x12_E000 – 0x12_FFFF | 8,192 | Reserved |
| 0x13_0000 – 0x13_EFFF | 61,440 | Reserved |
| 0x13_F000 – 0x13_FFFF | 4,096 | Buffer RAM (User and EEE) |

1. MMCCTL1 register bit

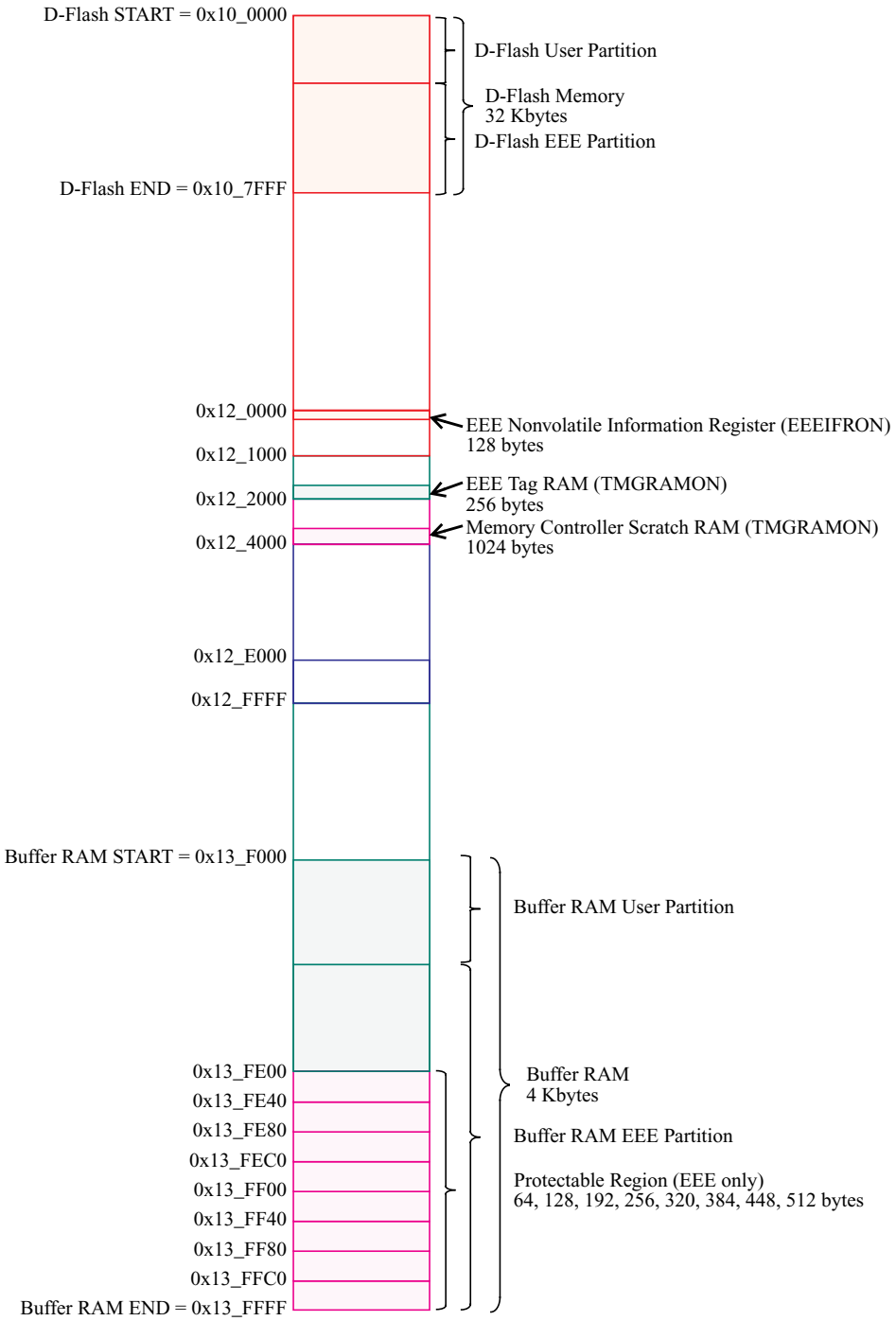



**Figure 28-3. EEE Resource Memory Map**

The Full Partition D-Flash command (see Section 28.4.2.15) is used to program the EEE nonvolatile information register fields where address 0x12_0000 defines the D-Flash partition for user access and address 0x12_0004 defines the buffer RAM partition for EEE operations.

**Table 28-7. EEE Nonvolatile Information Register Fields**

| Global Address (EEEIFRON) | Size (Bytes) | Description |
|---|---|---|
| 0x12_0000 – 0x12_0001 | 2 | D-Flash User Partition (DFPART)<br>Refer to Section 28.4.2.15, “Full Partition D-Flash Command” |
| 0x12_0002 – 0x12_0003 | 2 | D-Flash User Partition (duplicate(1)) |
| 0x12_0004 – 0x12_0005 | 2 | Buffer RAM EEE Partition (ERPART)<br>Refer to Section 28.4.2.15, “Full Partition D-Flash Command” |
| 0x12_0006 – 0x12_0007 | 2 | Buffer RAM EEE Partition (duplicate[1]) |
| 0x12_0008 – 0x12_007F | 120 | Reserved |

1. Duplicate value used if primary value generates a double bit fault when read during the reset sequence.

## 28.3.2 Register Descriptions

The Flash module contains a set of 20 control and status registers located between Flash module base + 0x0000 and 0x0013. A summary of the Flash module registers is given in Figure 28-4 with detailed descriptions in the following subsections.

**CAUTION**

Writes to any Flash register must be avoided while a Flash command is active (CCIF=0) to prevent corruption of Flash register contents and Memory Controller behavior.

| Address & Name | | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|
| 0x0000 FCLKDIV | R | FDIVLD | FDIV6 | FDIV5 | FDIV4 | FDIV3 | FDIV2 | FDIV1 | FDIV0 |
| | W | | | | | | | | |
| 0x0001 FSEC | R | KEYEN1 | KEYEN0 | RNV5 | RNV4 | RNV3 | RNV2 | SEC1 | SEC0 |
| | W | | | | | | | | |
| 0x0002 FCCOBIX | R | 0 | 0 | 0 | 0 | 0 | CCOBIX2 | CCOBIX1 | CCOBIX0 |
| | W | | | | | | | | |
| 0x0003 FECCRIX | R | 0 | 0 | 0 | 0 | 0 | ECCRIX2 | ECCRIX1 | ECCRIX0 |
| | W | | | | | | | | |
| 0x0004 FCNFG | R | CCIE | 0 | 0 | IGNSF | 0 | 0 | FDFD | FSFD |
| | W | | | | | | | | |

**Figure 28-4. FTM768K4 Register Summary**

| Address & Name | | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|
| 0x0005 FERCNFG | R | ERSERIE | PGMERIE | 0 | EPVIOLIE | ERSVIE1 | ERSVIE0 | DFDIE | SFDIE |
| | W | | | | | | | | |
| 0x0006 FSTAT | R | CCIF | 0 | ACCERR | FPVIOL | MGBUSY | RSVD | MGSTAT1 | MGSTAT0 |
| | W | | | | | | | | |
| 0x0007 FERSTAT | R | ERSERIF | PGMERIF | 0 | EPVIOLIF | ERSVIF1 | ERSVIF0 | DFDIF | SFDIF |
| | W | | | | | | | | |
| 0x0008 FPROT | R | FPOPEN | RNV6 | FPHDIS | FPHS1 | FPHS0 | FPLDIS | FPLS1 | FPLS0 |
| | W | | | | | | | | |
| 0x0009 EPROT | R | EPOPEN | RNV6 | RNV5 | RNV4 | EPDIS | EPS2 | EPS1 | EPS0 |
| | W | | | | | | | | |
| 0x000A FCCOBHI | R | CCOB15 | CCOB14 | CCOB13 | CCOB12 | CCOB11 | CCOB10 | CCOB9 | CCOB8 |
| | W | | | | | | | | |
| 0x000B FCCOBLO | R | CCOB7 | CCOB6 | CCOB5 | CCOB4 | CCOB3 | CCOB2 | CCOB1 | CCOB0 |
| | W | | | | | | | | |
| 0x000C ETAGHI | R | ETAG15 | ETAG14 | ETAG13 | ETAG12 | ETAG11 | ETAG10 | ETAG9 | ETAG8 |
| | W | | | | | | | | |
| 0x000D ETAGLO | R | ETAG7 | ETAG6 | ETAG5 | ETAG4 | ETAG3 | ETAG2 | ETAG1 | ETAG0 |
| | W | | | | | | | | |
| 0x000E FECCRHI | R | ECCR15 | ECCR14 | ECCR13 | ECCR12 | ECCR11 | ECCR10 | ECCR9 | ECCR8 |
| | W | | | | | | | | |
| 0x000F FECCRLO | R | ECCR7 | ECCR6 | ECCR5 | ECCR4 | ECCR3 | ECCR2 | ECCR1 | ECCR0 |
| | W | | | | | | | | |
| 0x0010 FOPT | R | NV7 | NV6 | NV5 | NV4 | NV3 | NV2 | NV1 | NV0 |
| | W | | | | | | | | |
| 0x0011 FRSV0 | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | W | | | | | | | | |
| 0x0012 FRSV1 | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | W | | | | | | | | |

**Figure 28-4. FTM768K4 Register Summary (continued)**

| Address & Name | | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|
| 0x0013 FRSV2 | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | W | | | | | | | | |

= Unimplemented or Reserved

**Figure 28-4. FTM768K4 Register Summary (continued)**

### 28.3.2.1 Flash Clock Divider Register (FCLKDIV)

The FCLKDIV register is used to control timed events in program and erase algorithms.

Offset Module Base + 0x0000

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | FDIVLD | FDIV[6:0] | | | | | | |
| W | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented or Reserved

**Figure 28-5. Flash Clock Divider Register (FCLKDIV)**

All bits in the FCLKDIV register are readable, bits 6–0 are write once and bit 7 is not writable.

**Table 28-8. FCLKDIV Field Descriptions**

| Field | Description |
|---|---|
| 7<br>FDIVLD | **Clock Divider Loaded**<br>0 FCLKDIV register has not been written<br>1 FCLKDIV register has been written since the last reset |
| 6–0<br>FDIV[6:0] | **Clock Divider Bits** — FDIV[6:0] must be set to effectively divide OSCCLK down to generate an internal Flash clock, FCLK, with a target frequency of 1 MHz for use by the Flash module to control timed events during program and erase algorithms. Table 28-9 shows recommended values for FDIV[6:0] based on OSCCLK frequency. Please refer to Section 28.4.1, “Flash Command Operations,” for more information. |

**CAUTION**

The FCLKDIV register should never be written while a Flash command is executing (CCIF=0). The FCLKDIV register is writable during the Flash reset sequence even though CCIF is clear.

**Table 28-9. FDIV vs OSCCLK Frequency**

| OSCCLK Frequency (MHz) | | FDIV[6:0] | OSCCLK Frequency (MHz) | | FDIV[6:0] | OSCCLK Frequency (MHz) | | FDIV[6:0] |
|---|---|---|---|---|---|---|---|---|
| MIN(1) | MAX(2) | | MIN1 | MAX2 | | MIN1 | MAX2 | |
| | | | 33.60 | 34.65 | 0x20 | 67.20 | 68.25 | 0x40 |
| 1.60 | 2.10 | 0x01 | 34.65 | 35.70 | 0x21 | 68.25 | 69.30 | 0x41 |
| 2.40 | 3.15 | 0x02 | 35.70 | 36.75 | 0x22 | 69.30 | 70.35 | 0x42 |
| 3.20 | 4.20 | 0x03 | 36.75 | 37.80 | 0x23 | 70.35 | 71.40 | 0x43 |
| 4.20 | 5.25 | 0x04 | 37.80 | 38.85 | 0x24 | 71.40 | 72.45 | 0x44 |
| 5.25 | 6.30 | 0x05 | 38.85 | 39.90 | 0x25 | 72.45 | 73.50 | 0x45 |
| 6.30 | 7.35 | 0x06 | 39.90 | 40.95 | 0x26 | 73.50 | 74.55 | 0x46 |
| 7.35 | 8.40 | 0x07 | 40.95 | 42.00 | 0x27 | 74.55 | 75.60 | 0x47 |
| 8.40 | 9.45 | 0x08 | 42.00 | 43.05 | 0x28 | 75.60 | 76.65 | 0x48 |
| 9.45 | 10.50 | 0x09 | 43.05 | 44.10 | 0x29 | 76.65 | 77.70 | 0x49 |
| 10.50 | 11.55 | 0x0A | 44.10 | 45.15 | 0x2A | 77.70 | 78.75 | 0x4A |
| 11.55 | 12.60 | 0x0B | 45.15 | 46.20 | 0x2B | 78.75 | 79.80 | 0x4B |
| 12.60 | 13.65 | 0x0C | 46.20 | 47.25 | 0x2C | 79.80 | 80.85 | 0x4C |
| 13.65 | 14.70 | 0x0D | 47.25 | 48.30 | 0x2D | 80.85 | 81.90 | 0x4D |
| 14.70 | 15.75 | 0x0E | 48.30 | 49.35 | 0x2E | 81.90 | 82.95 | 0x4E |
| 15.75 | 16.80 | 0x0F | 49.35 | 50.40 | 0x2F | 82.95 | 84.00 | 0x4F |
| 16.80 | 17.85 | 0x10 | 50.40 | 51.45 | 0x30 | 84.00 | 85.05 | 0x50 |
| 17.85 | 18.90 | 0x11 | 51.45 | 52.50 | 0x31 | 85.05 | 86.10 | 0x51 |
| 18.90 | 19.95 | 0x12 | 52.50 | 53.55 | 0x32 | 86.10 | 87.15 | 0x52 |
| 19.95 | 21.00 | 0x13 | 53.55 | 54.60 | 0x33 | 87.15 | 88.20 | 0x53 |
| 21.00 | 22.05 | 0x14 | 54.60 | 55.65 | 0x34 | 88.20 | 89.25 | 0x54 |
| 22.05 | 23.10 | 0x15 | 55.65 | 56.70 | 0x35 | 89.25 | 90.30 | 0x55 |
| 23.10 | 24.15 | 0x16 | 56.70 | 57.75 | 0x36 | 90.30 | 91.35 | 0x56 |
| 24.15 | 25.20 | 0x17 | 57.75 | 58.80 | 0x37 | 91.35 | 92.40 | 0x57 |
| 25.20 | 26.25 | 0x18 | 58.80 | 59.85 | 0x38 | 92.40 | 93.45 | 0x58 |
| 26.25 | 27.30 | 0x19 | 59.85 | 60.90 | 0x39 | 93.45 | 94.50 | 0x59 |
| 27.30 | 28.35 | 0x1A | 60.90 | 61.95 | 0x3A | 94.50 | 95.55 | 0x5A |
| 28.35 | 29.40 | 0x1B | 61.95 | 63.00 | 0x3B | 95.55 | 96.60 | 0x5B |
| 29.40 | 30.45 | 0x1C | 63.00 | 64.05 | 0x3C | 96.60 | 97.65 | 0x5C |
| 30.45 | 31.50 | 0x1D | 64.05 | 65.10 | 0x3D | 97.65 | 98.70 | 0x5D |
| 31.50 | 32.55 | 0x1E | 65.10 | 66.15 | 0x3E | 98.70 | 99.75 | 0x5E |
| 32.55 | 33.60 | 0x1F | 66.15 | 67.20 | 0x3F | 99.75 | 100.80 | 0x5F |

1. FDIV shown generates an FCLK frequency of >0.8 MHz

2. FDIV shown generates an FCLK frequency of 1.05 MHz

### 28.3.2.2 Flash Security Register (FSEC)

The FSEC register holds all bits associated with the security of the MCU and Flash module.

Offset Module Base + 0x0001

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | KEYEN[1:0] | | RNV[5:2] | | | | SEC[1:0] | |
| W | | | | | | | | |
| Reset | F | F | F | F | F | F | F | F |

= Unimplemented or Reserved

**Figure 28-6. Flash Security Register (FSEC)**

All bits in the FSEC register are readable but not writable.

During the reset sequence, the FSEC register is loaded with the contents of the Flash security byte in the Flash configuration field at global address 0x7F_FF0F located in P-Flash memory (see Table 28-3) as indicated by reset condition F in Figure 28-6. If a double bit fault is detected while reading the P-Flash phrase containing the Flash security byte during the reset sequence, all bits in the FSEC register will be set to leave the Flash module in a secured state with backdoor key access disabled.

**Table 28-10. FSEC Field Descriptions**

| Field | Description |
|---|---|
| 7–6<br>KEYEN[1:0] | **Backdoor Key Security Enable Bits** — The KEYEN[1:0] bits define the enabling of backdoor key access to the Flash module as shown in Table 28-11. |
| 5–2<br>RNV[5:2} | **Reserved Nonvolatile Bits** — The RNV bits should remain in the erased state for future enhancements. |
| 1–0<br>SEC[1:0] | **Flash Security Bits** — The SEC[1:0] bits define the security state of the MCU as shown in Table 28-12. If the Flash module is unsecured using backdoor key access, the SEC bits are forced to 10. |

**Table 28-11. Flash KEYEN States**

| KEYEN[1:0] | Status of Backdoor Key Access |
|---|---|
| 00 | DISABLED |
| 01 | DISABLED(1) |
| 10 | ENABLED |
| 11 | DISABLED |

1. Preferred KEYEN state to disable backdoor key access.

**Table 28-12. Flash Security States**

| SEC[1:0] | Status of Security |
|---|---|
| 00 | SECURED |
| 01 | SECURED(1) |
| 10 | UNSECURED |
| 11 | SECURED |

1. Preferred SEC state to set MCU to secured state.

The security function in the Flash module is described in Section 28.5.

### 28.3.2.3 Flash CCOB Index Register (FCCOBIX)

The FCCOBIX register is used to index the FCCOB register for Flash memory operations.

Offset Module Base + 0x0002

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | 0 | 0 | 0 | CCOBIX[2:0] | | |
| W | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented or Reserved

**Figure 28-7. FCCOB Index Register (FCCOBIX)**

CCOBIX bits are readable and writable while remaining bits read 0 and are not writable.

**Table 28-13. FCCOBIX Field Descriptions**

| Field | Description |
|---|---|
| 2–0 CCOBIX[1:0] | **Common Command Register Index**— The CCOBIX bits are used to select which word of the FCCOB register array is being read or written to. See Section 28.3.2.11, “Flash Common Command Object Register (FCCOB),” for more details. |

### 28.3.2.4 Flash ECCR Index Register (FECCRIX)

The FECCRIX register is used to index the FECCR register for ECC fault reporting.

Offset Module Base + 0x0003

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | 0 | 0 | 0 | ECCRIX[2:0] | | |
| W | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented or Reserved

**Figure 28-8. FECCR Index Register (FECCRIX)**

ECCRIX bits are readable and writable while remaining bits read 0 and are not writable.

**Table 28-14. FECCRIX Field Descriptions**

| Field | Description |
|---|---|
| 2-0<br>ECCRIX[2:0] | **ECC Error Register Index**— The ECCRIX bits are used to select which word of the FECCR register array is being read. See Section 28.3.2.13, “Flash ECC Error Results Register (FECCR),” for more details. |

### 28.3.2.5 Flash Configuration Register (FCNFG)

The FCNFG register enables the Flash command complete interrupt and forces ECC faults on Flash array read access from the CPU or XGATE.

Offset Module Base + 0x0004

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | CCIE | 0 | 0 | IGNSF | 0 | 0 | FDFD | FSFD |
| W | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented or Reserved

**Figure 28-9. Flash Configuration Register (FCNFG)**

CCIE, IGNSF, FDFD, and FSFD bits are readable and writable while remaining bits read 0 and are not writable.

**Table 28-15. FCNFG Field Descriptions**

| Field | Description |
|---|---|
| 7<br>CCIE | **Command Complete Interrupt Enable** — The CCIE bit controls interrupt generation when a Flash command has completed.<br>0 Command complete interrupt disabled<br>1 An interrupt will be requested whenever the CCIF flag in the FSTAT register is set (see Section 28.3.2.7) |
| 4<br>IGNSF | **Ignore Single Bit Fault** — The IGNSF controls single bit fault reporting in the FERSTAT register (see Section 28.3.2.8).<br>0 All single bit faults detected during array reads are reported<br>1 Single bit faults detected during array reads are not reported and the single bit fault interrupt will not be generated |

**Table 28-15. FCNFG Field Descriptions (continued)**

| Field | Description |
|---|---|
| 1<br>FDFD | **Force Double Bit Fault Detect** — The FDFD bit allows the user to simulate a double bit fault during Flash array read operations and check the associated interrupt routine. The FDFD bit is cleared by writing a 0 to FDFD. The FECCR registers will not be updated during the Flash array read operation with FDFD set unless an actual double bit fault is detected.<br>0 Flash array read operations will set the DFDIF flag in the FERSTAT register only if a double bit fault is detected<br>1 Any Flash array read operation will force the DFDIF flag in the FERSTAT register to be set (see Section 28.3.2.7) and an interrupt will be generated as long as the DFDIE interrupt enable in the FERCNFG register is set (see Section 28.3.2.6) |
| 0<br>FSFD | **Force Single Bit Fault Detect** — The FSFD bit allows the user to simulate a single bit fault during Flash array read operations and check the associated interrupt routine. The FSFD bit is cleared by writing a 0 to FSFD. The FECCR registers will not be updated during the Flash array read operation with FSFD set unless an actual single bit fault is detected.<br>0 Flash array read operations will set the SFDIF flag in the FERSTAT register only if a single bit fault is detected<br>1 Flash array read operation will force the SFDIF flag in the FERSTAT register to be set (see Section 28.3.2.7) and an interrupt will be generated as long as the SFDIE interrupt enable in the FERCNFG register is set (see Section 28.3.2.6) |

## 28.3.2.6 Flash Error Configuration Register (FERCNFG)

The FERCNFG register enables the Flash error interrupts for the FERSTAT flags.

Offset Module Base + 0x0005

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | ERSERIE | PGMERIE | 0 | EPVIOLIE | ERSVIE1 | ERSVIE0 | DFDIE | SFDIE |
| W | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented or Reserved

**Figure 28-10. Flash Error Configuration Register (FERCNFG)**

All assigned bits in the FERCNFG register are readable and writable.

**Table 28-16. FERCNFG Field Descriptions**

| Field | Description |
|---|---|
| 7<br>ERSERIE | **EEE Erase Error Interrupt Enable** — The ERSERIE bit controls interrupt generation when a failure is detected during an EEE erase operation.<br>0 ERSERIF interrupt disabled<br>1 An interrupt will be requested whenever the ERSERIF flag is set (see Section 28.3.2.8) |
| 6<br>PGMERIE | **EEE Program Error Interrupt Enable** — The PGMERIE bit controls interrupt generation when a failure is detected during an EEE program operation.<br>0 PGMERIF interrupt disabled<br>1 An interrupt will be requested whenever the PGMERIF flag is set (see Section 28.3.2.8) |
| 4<br>EPVIOLIE | **EEE Protection Violation Interrupt Enable** — The EPVIOLIE bit controls interrupt generation when a protection violation is detected during a write to the buffer RAM EEE partition.<br>0 EPVIOLIF interrupt disabled<br>1 An interrupt will be requested whenever the EPVIOLIF flag is set (see Section 28.3.2.8) |

**Table 28-16. FERCNFG Field Descriptions (continued)**

| Field | Description |
|---|---|
| 3<br>ERSVIE1 | **EEE Error Type 1 Interrupt Enable** — The ERSVIE1 bit controls interrupt generation when a change state error is detected during an EEE operation.<br>0 ERSVIF1 interrupt disabled<br>1 An interrupt will be requested whenever the ERSVIF1 flag is set (see Section 28.3.2.8) |
| 2<br>ERSVIE0 | **EEE Error Type 0 Interrupt Enable** — The ERSVIE0 bit controls interrupt generation when a sector format error is detected during an EEE operation.<br>0 ERSVIF0 interrupt disabled<br>1 An interrupt will be requested whenever the ERSVIF0 flag is set (see Section 28.3.2.8) |
| 1<br>DFDIE | **Double Bit Fault Detect Interrupt Enable** — The DFDIE bit controls interrupt generation when a double bit fault is detected during a Flash block read operation.<br>0 DFDIF interrupt disabled<br>1 An interrupt will be requested whenever the DFDIF flag is set (see Section 28.3.2.8) |
| 0<br>SFDIE | **Single Bit Fault Detect Interrupt Enable** — The SFDIE bit controls interrupt generation when a single bit fault is detected during a Flash block read operation.<br>0 SFDIF interrupt disabled whenever the SFDIF flag is set (see Section 28.3.2.8)<br>1 An interrupt will be requested whenever the SFDIF flag is set (see Section 28.3.2.8) |

## 28.3.2.7 Flash Status Register (FSTAT)

The FSTAT register reports the operational status of the Flash module.

Offset Module Base + 0x0006

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | CCIF | 0 | ACCERR | FPVIOL | MGBUSY | RSVD | MGSTAT[1:0] | |
| W | | | | | | | | |
| Reset | 1 | 0 | 0 | 0 | 0 | 0 | 0(1) | 0[1] |

= Unimplemented or Reserved

**Figure 28-11. Flash Status Register (FSTAT)**

1. Reset value can deviate from the value shown if a double bit fault is detected during the reset sequence (see Section 28.6).

CCIF, ACCERR, and FPVIOL bits are readable and writable, MGBUSY and MGSTAT bits are readable but not writable, while remaining bits read 0 and are not writable.

**Table 28-17. FSTAT Field Descriptions**

| Field | Description |
|---|---|
| 7<br>CCIF | **Command Complete Interrupt Flag** — The CCIF flag indicates that a Flash command has completed. The CCIF flag is cleared by writing a 1 to CCIF to launch a command and CCIF will stay low until command completion or command violation.<br>0 Flash command in progress<br>1 Flash command has completed |
| 5<br>ACCERR | **Flash Access Error Flag** — The ACCERR bit indicates an illegal access has occurred to the Flash memory caused by either a violation of the command write sequence (see Section 28.4.1.2) or issuing an illegal Flash command or when errors are encountered while initializing the EEE buffer ram during the reset sequence. While ACCERR is set, the CCIF flag cannot be cleared to launch a command. The ACCERR bit is cleared by writing a 1 to ACCERR. Writing a 0 to the ACCERR bit has no effect on ACCERR.<br>0 No access error detected<br>1 Access error detected |
| 4<br>FPVIOL | **Flash Protection Violation Flag** —The FPVIOL bit indicates an attempt was made to program or erase an address in a protected area of P-Flash memory during a command write sequence. The FPVIOL bit is cleared by writing a 1 to FPVIOL. Writing a 0 to the FPVIOL bit has no effect on FPVIOL. While FPVIOL is set, it is not possible to launch a command or start a command write sequence.<br>0 No protection violation detected<br>1 Protection violation detected |
| 3<br>MGBUSY | **Memory Controller Busy Flag** — The MGBUSY flag reflects the active state of the Memory Controller.<br>0 Memory Controller is idle<br>1 Memory Controller is busy executing a Flash command (CCIF = 0) or is handling internal EEE operations |
| 2<br>RSVD | **Reserved Bit** — This bit is reserved and always reads 0. |
| 1–0<br>MGSTAT[1:0] | **Memory Controller Command Completion Status Flag** — One or more MGSTAT flag bits are set if an error is detected during execution of a Flash command or during the Flash reset sequence. See Section 28.4.2, “Flash Command Description,” and Section 28.6, “Initialization” for details. |

### 28.3.2.8 Flash Error Status Register (FERSTAT)

The FERSTAT register reflects the error status of internal Flash operations.

Offset Module Base + 0x0007

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | ERSERIF | PGMERIF | 0 | EPVIOLIF | ERSVIF1 | ERSVIF0 | DFDIF | SFDIF |
| W | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented or Reserved

**Figure 28-12. Flash Error Status Register (FERSTAT)**

All flags in the FERSTAT register are readable and only writable to clear the flag.

**Table 28-18. FERSTAT Field Descriptions**

| Field | Description |
|---|---|
| 7<br>ERSERIF | **EEE Erase Error Interrupt Flag** — The setting of the ERSERIF flag occurs due to an error in a Flash erase command that resulted in the erase operation not being successful during EEE operations. The ERSERIF flag is cleared by writing a 1 to ERSERIF. Writing a 0 to the ERSERIF flag has no effect on ERSERIF. While ERSERIF is set, it is possible to write to the buffer RAM EEE partition but the data written will not be transferred to the D-Flash EEE partition.<br>0 Erase command successfully completed on the D-Flash EEE partition<br>1 Erase command failed on the D-Flash EEE partition |
| 6<br>PGMERIF | **EEE Program Error Interrupt Flag** — The setting of the PGMERIF flag occurs due to an error in a Flash program command that resulted in the program operation not being successful during EEE operations. The PGMERIF flag is cleared by writing a 1 to PGMERIF. Writing a 0 to the PGMERIF flag has no effect on PGMERIF. While PGMERIF is set, it is possible to write to the buffer RAM EEE partition but the data written will not be transferred to the D-Flash EEE partition.<br>0 Program command successfully completed on the D-Flash EEE partition<br>1 Program command failed on the D-Flash EEE partition |
| 4<br>EPVIOLIF | **EEE Protection Violation Interrupt Flag** —The setting of the EPVIOLIF flag indicates an attempt was made to write to a protected area of the buffer RAM EEE partition. The EPVIOLIF flag is cleared by writing a 1 to EPVIOLIF. Writing a 0 to the EPVIOLIF flag has no effect on EPVIOLIF. While EPVIOLIF is set, it is possible to write to the buffer RAM EEE partition as long as the address written to is not in a protected area.<br>0 No EEE protection violation<br>1 EEE protection violation detected |
| 3<br>ERSVIF1 | **EEE Error Interrupt 1 Flag** —The setting of the ERSVIF1 flag indicates that the memory controller was unable to change the state of a D-Flash EEE sector. The ERSVIF1 flag is cleared by writing a 1 to ERSVIF1. Writing a 0 to the ERSVIF1 flag has no effect on ERSVIF1. While ERSVIF1 is set, it is possible to write to the buffer RAM EEE partition but the data written will not be transferred to the D-Flash EEE partition.<br>0 No EEE sector state change error detected<br>1 EEE sector state change error detected |
| 2<br>ERSVIF0 | **EEE Error Interrupt 0 Flag** —The setting of the ERSVIF0 flag indicates that the memory controller was unable to format a D-Flash EEE sector for EEE use. The ERSVIF0 flag is cleared by writing a 1 to ERSVIF0. Writing a 0 to the ERSVIF0 flag has no effect on ERSVIF0. While ERSVIF0 is set, it is possible to write to the buffer RAM EEE partition but the data written will not be transferred to the D-Flash EEE partition.<br>0 No EEE sector format error detected<br>1 EEE sector format error detected |
| 1<br>DFDIF | **Double Bit Fault Detect Interrupt Flag** — The setting of the DFDIF flag indicates that a double bit fault was detected in the stored parity and data bits during a Flash array read operation or that a Flash array read operation was attempted on a Flash block that was under a Flash command operation. The DFDIF flag is cleared by writing a 1 to DFDIF. Writing a 0 to DFDIF has no effect on DFDIF.<br>0 No double bit fault detected<br>1 Double bit fault detected or an invalid Flash array read operation attempted |
| 0<br>SFDIF | **Single Bit Fault Detect Interrupt Flag** — With the IGNSF bit in the FCNFG register clear, the SFDIF flag indicates that a single bit fault was detected in the stored parity and data bits during a Flash array read operation or that a Flash array read operation was attempted on a Flash block that was under a Flash command operation. The SFDIF flag is cleared by writing a 1 to SFDIF. Writing a 0 to SFDIF has no effect on SFDIF.<br>0 No single bit fault detected<br>1 Single bit fault detected and corrected or an invalid Flash array read operation attempted |

### 28.3.2.9 P-Flash Protection Register (FPROT)

The FPROT register defines which P-Flash sectors are protected against program and erase operations.

Offset Module Base + 0x0008

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | FPOPEN | RNV6 | FPHDIS | FPHS[1:0] | | FPLDIS | FPLS[1:0] | |
| W | | | | | | | | |
| Reset | F | F | F | F | F | F | F | F |

= Unimplemented or Reserved

**Figure 28-13. Flash Protection Register (FPROT)**

The (unreserved) bits of the FPROT register are writable with the restriction that the size of the protected region can only be increased (see Section 28.3.2.9.1, “P-Flash Protection Restrictions,” and Table 28-23).

During the reset sequence, the FPROT register is loaded with the contents of the P-Flash protection byte in the Flash configuration field at global address 0x7F_FF0C located in P-Flash memory (see Table 28-3) as indicated by reset condition ‘F’ in Figure 28-13. To change the P-Flash protection that will be loaded during the reset sequence, the upper sector of the P-Flash memory must be unprotected, then the P-Flash protection byte must be reprogrammed. If a double bit fault is detected while reading the P-Flash phrase containing the P-Flash protection byte during the reset sequence, the FPOPEN bit will be cleared and remaining bits in the FPROT register will be set to leave the P-Flash memory fully protected.

Trying to alter data in any protected area in the P-Flash memory will result in a protection violation error and the FPVIOL bit will be set in the FSTAT register. The block erase of a P-Flash block is not possible if any of the P-Flash sectors contained in the same P-Flash block are protected.

**Table 28-19. FPROT Field Descriptions**

| Field | Description |
|---|---|
| 7<br>FPOPEN | **Flash Protection Operation Enable** — The FPOPEN bit determines the protection function for program or erase operations as shown in Table 28-20 for the P-Flash block.<br>0 When FPOPEN is clear, the FPHDIS and FPLDIS bits define unprotected address ranges as specified by the corresponding FPHS and FPLS bits<br>1 When FPOPEN is set, the FPHDIS and FPLDIS bits enable protection for the address range specified by the corresponding FPHS and FPLS bits |
| 6<br>RNV[6] | **Reserved Nonvolatile Bit** — The RNV bit should remain in the erased state for future enhancements. |
| 5<br>FPHDIS | **Flash Protection Higher Address Range Disable** — The FPHDIS bit determines whether there is a protected/unprotected area in a specific region of the P-Flash memory ending with global address 0x7F_FFFF.<br>0 Protection/Unprotection enabled<br>1 Protection/Unprotection disabled |
| 4–3<br>FPHS[1:0] | **Flash Protection Higher Address Size** — The FPHS bits determine the size of the protected/unprotected area in P-Flash memory as shown inTable 28-21. The FPHS bits can only be written to while the FPHDIS bit is set. |
| 2<br>FPLDIS | **Flash Protection Lower Address Range Disable** — The FPLDIS bit determines whether there is a protected/unprotected area in a specific region of the P-Flash memory beginning with global address 0x7F_8000.<br>0 Protection/Unprotection enabled<br>1 Protection/Unprotection disabled |
| 1–0<br>FPLS[1:0] | **Flash Protection Lower Address Size** — The FPLS bits determine the size of the protected/unprotected area in P-Flash memory as shown in Table 28-22. The FPLS bits can only be written to while the FPLDIS bit is set. |

**Table 28-20. P-Flash Protection Function**

| FPOPEN | FPHDIS | FPLDIS | Function[1] |
|---|---|---|---|
| 1 | 1 | 1 | No P-Flash Protection |
| 1 | 1 | 0 | Protected Low Range |
| 1 | 0 | 1 | Protected High Range |
| 1 | 0 | 0 | Protected High and Low Ranges |
| 0 | 1 | 1 | Full P-Flash Memory Protected |
| 0 | 1 | 0 | Unprotected Low Range |
| 0 | 0 | 1 | Unprotected High Range |
| 0 | 0 | 0 | Unprotected High and Low Ranges |

1. For range sizes, refer to Table 28-21 and Table 28-22.

**Table 28-21. P-Flash Protection Higher Address Range**

| FPHS[1:0] | Global Address Range | Protected Size |
|---|---|---|
| 00 | 0x7F_F800–0x7F_FFFF | 2 Kbytes |
| 01 | 0x7F_F000–0x7F_FFFF | 4 Kbytes |
| 10 | 0x7F_E000–0x7F_FFFF | 8 Kbytes |
| 11 | 0x7F_C000–0x7F_FFFF | 16 Kbytes |

**Table 28-22. P-Flash Protection Lower Address Range**

| FPLS[1:0] | Global Address Range | Protected Size |
|---|---|---|
| 00 | 0x7F_8000–0x7F_83FF | 1 Kbyte |
| 01 | 0x7F_8000–0x7F_87FF | 2 Kbytes |
| 10 | 0x7F_8000–0x7F_8FFF | 4 Kbytes |
| 11 | 0x7F_8000–0x7F_9FFF | 8 Kbytes |

All possible P-Flash protection scenarios are shown in Figure 28-14. Although the protection scheme is loaded from the Flash memory at global address 0x7F_FF0C during the reset sequence, it can be changed by the user. The P-Flash protection scheme can be used by applications requiring reprogramming in single chip mode while providing as much protection as possible if reprogramming is not required.

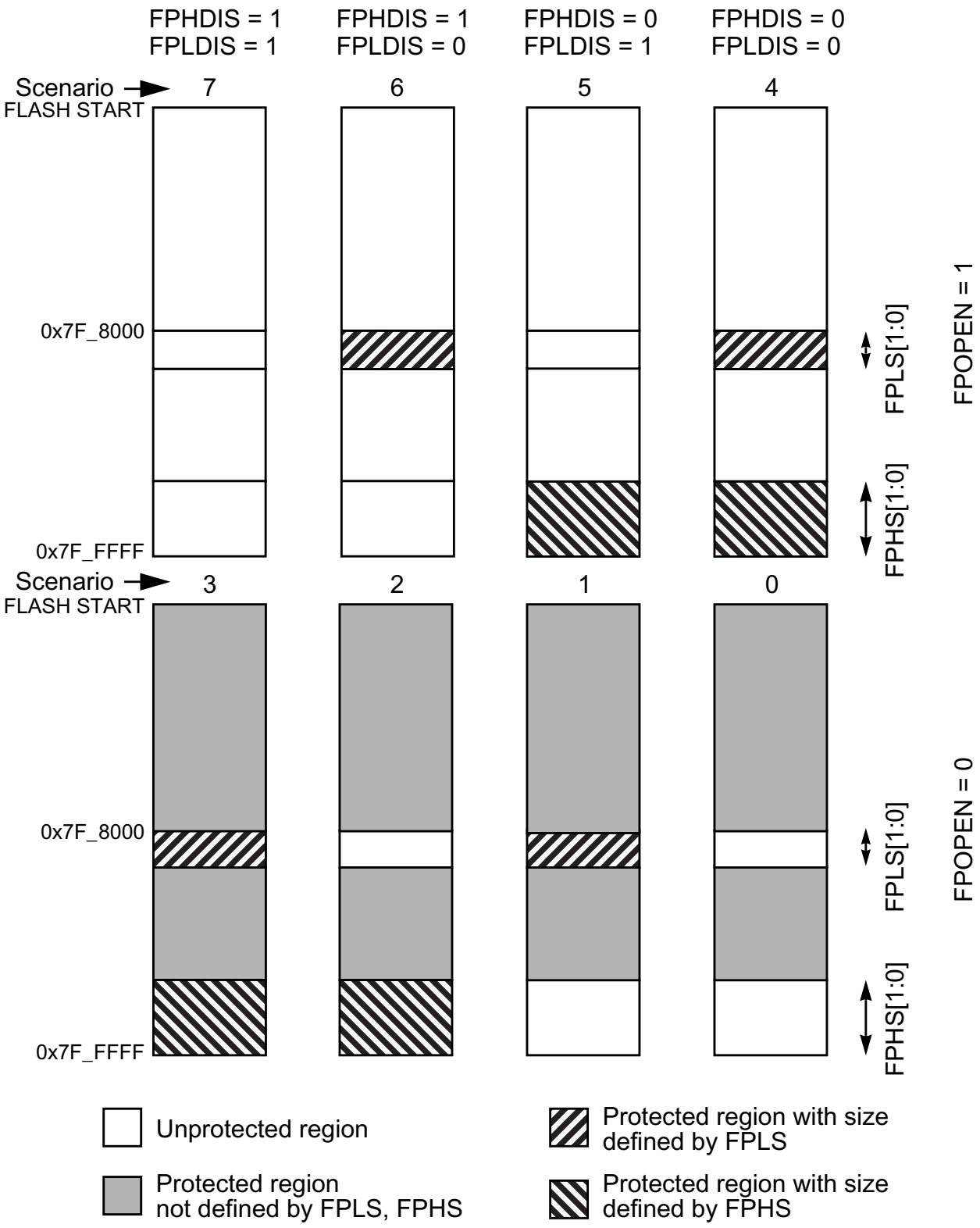



**Figure 28-14. P-Flash Protection Scenarios**

#### 28.3.2.9.1 P-Flash Protection Restrictions

The general guideline is that P-Flash protection can only be added and not removed. Table 28-23 specifies all valid transitions between P-Flash protection scenarios. Any attempt to write an invalid scenario to the FPROT register will be ignored. The contents of the FPROT register reflect the active protection scenario. See the FPHS and FPLS bit descriptions for additional restrictions.

**Table 28-23. P-Flash Protection Scenario Transitions**

| From Protection Scenario | To Protection Scenario[1] | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| 0 | X | X | X | X | | | | |
| 1 | | X | | X | | | | |
| 2 | | | X | X | | | | |
| 3 | | | | X | | | | |
| 4 | | | | X | X | | | |
| 5 | | | X | X | X | X | | |
| 6 | | X | | X | X | | X | |
| 7 | X | X | X | X | X | X | X | X |

1. Allowed transitions marked with X, see Figure 28-14 for a definition of the scenarios.

### 28.3.2.10 EEE Protection Register (EPROT)

The EPROT register defines which buffer RAM EEE partition areas are protected against writes.

Offset Module Base + 0x0009

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | EPOPEN | RNV[6:4] | | | EPDIS | EPS[2:0] | | |
| W | | | | | | | | |
| Reset | F | F | F | F | F | F | F | F |

= Unimplemented or Reserved

**Figure 28-15. EEE Protection Register (EPROT)**

All bits in the EPROT register are readable and writable except for RNV[6:4] which are only readable. The EPOPEN and EPDIS bits can only be written to the protected state. The EPS bits can be written anytime until the EPDIS bit is cleared. If the EPOPEN bit is cleared, the state of the EPDIS and EPS bits is irrelevant.

During the reset sequence, the EPROT register is loaded from the EEE protection byte in the Flash configuration field at global address 0x7F_FF0D located in P-Flash memory (see Table 28-3) as indicated by reset condition F in Figure 28-15. To change the EEE protection that will be loaded during the reset sequence, the P-Flash sector containing the EEE protection byte must be unprotected, then the EEE protection byte must be programmed. If a double bit fault is detected while reading the P-Flash phrase

containing the EEE protection byte during the reset sequence, the EPOPEN bit will be cleared and remaining bits in the EPROT register will be set to leave the buffer RAM EEE partition fully protected.

Trying to write data to any protected area in the buffer RAM EEE partition will result in a protection violation error and the EPVIOLIF flag will be set in the FERSTAT register. Trying to write data to any protected area in the buffer RAM partitioned for user access will not be prevented and the EPVIOLIF flag in the FERSTAT register will not set.

**Table 28-24. EPROT Field Descriptions**

| Field | Description |
|---|---|
| 7<br>EPOPEN | **Enables writes to the Buffer RAM partitioned for EEE**<br>0 The entire buffer RAM EEE partition is protected from writes<br>1 Unprotected buffer RAM EEE partition areas are enabled for writes |
| 6–4<br>RNV[6:4] | **Reserved Nonvolatile Bits** — The RNV bits should remain in the erased state for future enhancements |
| 3<br>EPDIS | **Buffer RAM Protection Address Range Disable** — The EPDIS bit determines whether there is a protected area in a specific region of the buffer RAM EEE partition.<br>0 Protection enabled<br>1 Protection disabled |
| 2–0<br>EPS[2:0] | **Buffer RAM Protection Size** — The EPS[2:0] bits determine the size of the protected area in the buffer RAM EEE partition as shown inTable 28-21. The EPS bits can only be written to while the EPDIS bit is set. |

**Table 28-25. Buffer RAM EEE Partition Protection Address Range**

| EPS[2:0] | Global Address Range | Protected Size |
|---|---|---|
| 000 | 0x13_FFC0 – 0x13_FFFF | 64 bytes |
| 001 | 0x13_FF80 – 0x13_FFFF | 128 bytes |
| 010 | 0x13_FF40 – 0x13_FFFF | 192 bytes |
| 011 | 0x13_FF00 – 0x13_FFFF | 256 bytes |
| 100 | 0x13_FEC0 – 0x13_FFFF | 320 bytes |
| 101 | 0x13_FE80 – 0x13_FFFF | 384 bytes |
| 110 | 0x13_FE40 – 0x13_FFFF | 448 bytes |
| 111 | 0x13_FE00 – 0x13_FFFF | 512 bytes |

### 28.3.2.11 Flash Common Command Object Register (FCCOB)

The FCCOB is an array of six words addressed via the CCOBIX index found in the FCCOBIX register. Byte wide reads and writes are allowed to the FCCOB register.

Offset Module Base + 0x000A

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | CCOB[15:8] | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 28-16. Flash Common Command Object High Register (FCCOBHI)**

Offset Module Base + 0x000B

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | CCOB[7:0] | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 28-17. Flash Common Command Object Low Register (FCCOBLO)**

### 28.3.2.11.1 FCCOB - NVM Command Mode

NVM command mode uses the indexed FCCOB register to provide a command code and its relevant parameters to the Memory Controller. The user first sets up all required FCCOB fields and then initiates the command's execution by writing a 1 to the CCIF bit in the FSTAT register (a 1 written by the user clears the CCIF command completion flag to 0). When the user clears the CCIF bit in the FSTAT register all FCCOB parameter fields are locked and cannot be changed by the user until the command completes (as evidenced by the Memory Controller returning CCIF to 1). Some commands return information to the FCCOB register array.

The generic format for the FCCOB parameter fields in NVM command mode is shown in Table 28-26. The return values are available for reading after the CCIF flag in the FSTAT register has been returned to 1 by the Memory Controller. Writes to the unimplemented parameter fields (CCOBIX = 110 and CCOBIX = 111) are ignored with reads from these fields returning 0x0000.

Table 28-26 shows the generic Flash command format. The high byte of the first word in the CCOB array contains the command code, followed by the parameters for this specific Flash command. For details on the FCCOB settings required by each command, see the Flash command descriptions in Section 28.4.2.

**Table 28-26. FCCOB - NVM Command Mode (Typical Usage)**

| CCOBIX[2:0] | Byte | FCCOB Parameter Fields (NVM Command Mode) |
|---|---|---|
| 000 | HI | FCMD[7:0] defining Flash command |
| | LO | 0, Global address [22:16] |
| 001 | HI | Global address [15:8] |
| | LO | Global address [7:0] |
| 010 | HI | Data 0 [15:8] |
| | LO | Data 0 [7:0] |

**Table 28-26. FCCOB - NVM Command Mode (Typical Usage)**

| CCOBIX[2:0] | Byte | FCCOB Parameter Fields (NVM Command Mode) |
|---|---|---|
| 011 | HI | Data 1 [15:8] |
| | LO | Data 1 [7:0] |
| 100 | HI | Data 2 [15:8] |
| | LO | Data 2 [7:0] |
| 101 | HI | Data 3 [15:8] |
| | LO | Data 3 [7:0] |

### 28.3.2.12 EEE Tag Counter Register (ETAG)

The ETAG register contains the number of outstanding words in the buffer RAM EEE partition that need to be programmed into the D-Flash EEE partition. The ETAG register is decremented prior to the related tagged word being programmed into the D-Flash EEE partition. All tagged words have been programmed into the D-Flash EEE partition once all bits in the ETAG register read 0 and the MGBUSY flag in the FSTAT register reads 0.

Offset Module Base + 0x000C

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | ETAG[15:8] | | | | | | | |
| W | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented or Reserved

**Figure 28-18. EEE Tag Counter High Register (ETAGHI)**

Offset Module Base + 0x000D

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | ETAG[7:0] | | | | | | | |
| W | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented or Reserved

**Figure 28-19. EEE Tag Counter Low Register (ETAGLO)**

All ETAG bits are readable but not writable and are cleared by the Memory Controller.

### 28.3.2.13 Flash ECC Error Results Register (FECCR)

The FECCR registers contain the result of a detected ECC fault for both single bit and double bit faults. The FECCR register provides access to several ECC related fields as defined by the ECCRIX index bits in the FECCRIX register (see Section 28.3.2.4). Once ECC fault information has been stored, no other

fault information will be recorded until the specific ECC fault flag has been cleared. In the event of simultaneous ECC faults, the priority for fault recording is:

1. Double bit fault over single bit fault
2. CPU over XGATE

Offset Module Base + 0x000E

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | ECCR[15:8] | | | | | | | |
| W | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented or Reserved

**Figure 28-20. Flash ECC Error Results High Register (FECCRHI)**

Offset Module Base + 0x000F

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | ECCR[7:0] | | | | | | | |
| W | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented or Reserved

**Figure 28-21. Flash ECC Error Results Low Register (FECCRLO)**

All FECCR bits are readable but not writable.

**Table 28-27. FECCR Index Settings**

| ECCRIX[2:0] | FECCR Register Content | | |
|---|---|---|---|
| | **Bits [15:8]** | **Bit[7]** | **Bits[6:0]** |
| 000 | Parity bits read from Flash block | CPU or XGATE source identity | Global address [22:16] |
| 001 | Global address [15:0] | | |
| 010 | Data 0 [15:0] | | |
| 011 | Data 1 [15:0] (P-Flash only) | | |
| 100 | Data 2 [15:0] (P-Flash only) | | |
| 101 | Data 3 [15:0] (P-Flash only) | | |
| 110 | Not used, returns 0x0000 when read | | |
| 111 | Not used, returns 0x0000 when read | | |

**Table 28-28. FECCR Index=000 Bit Descriptions**

| Field | Description |
|---|---|
| 15:8<br>PAR[7:0] | **ECC Parity Bits** — Contains the 8 parity bits from the 72 bit wide P-Flash data word or the 6 parity bits, allocated to PAR[5:0], from the 22 bit wide D-Flash word with PAR[7:6]=00. |
| 7<br>XBUS01 | **Bus Source Identifier** — The XBUS01 bit determines whether the ECC error was caused by a read access from the CPU or XGATE.<br>0 ECC Error happened on the CPU access<br>1 ECC Error happened on the XGATE access |
| 6–0<br>GADDR[22:16] | **Global Address** — The GADDR[22:16] field contains the upper seven bits of the global address having caused the error. |

The P-Flash word addressed by ECCRIX = 001 contains the lower 16 bits of the global address. The following four words addressed by ECCRIX = 010 to 101 contain the 64-bit wide data phrase. The four data words and the parity byte are the uncorrected data read from the P-Flash block.

The D-Flash word addressed by ECCRIX = 001 contains the lower 16 bits of the global address. The uncorrected 16-bit data word is addressed by ECCRIX = 010.

### 28.3.2.14 Flash Option Register (FOPT)

The FOPT register is the Flash option register.

Offset Module Base + 0x0010

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | NV[7:0] | | | | | | | |
| W | | | | | | | | |
| Reset | F | F | F | F | F | F | F | F |

= Unimplemented or Reserved

**Figure 28-22. Flash Option Register (FOPT)**

All bits in the FOPT register are readable but are not writable.

During the reset sequence, the FOPT register is loaded from the Flash nonvolatile byte in the Flash configuration field at global address 0x7F_FF0E located in P-Flash memory (see Table 28-3) as indicated by reset condition F in Figure 28-22. If a double bit fault is detected while reading the P-Flash phrase containing the Flash nonvolatile byte during the reset sequence, all bits in the FOPT register will be set.

**Table 28-29. FOPT Field Descriptions**

| Field | Description |
|---|---|
| 7–0<br>NV[7:0] | **Nonvolatile Bits** — The NV[7:0] bits are available as nonvolatile bits. Refer to the device user guide for proper use of the NV bits. |

### 28.3.2.15 Flash Reserved0 Register (FRSV0)

This Flash register is reserved for factory testing.

Offset Module Base + 0x0011

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| W | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented or Reserved

**Figure 28-23. Flash Reserved0 Register (FRSV0)**

All bits in the FRSV0 register read 0 and are not writable.

### 28.3.2.16 Flash Reserved1 Register (FRSV1)

This Flash register is reserved for factory testing.

Offset Module Base + 0x0012

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| W | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented or Reserved

**Figure 28-24. Flash Reserved1 Register (FRSV1)**

All bits in the FRSV1 register read 0 and are not writable.

### 28.3.2.17 Flash Reserved2 Register (FRSV2)

This Flash register is reserved for factory testing.

Offset Module Base + 0x0013

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| W | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented or Reserved

**Figure 28-25. Flash Reserved2 Register (FRSV2)**

All bits in the FRSV2 register read 0 and are not writable.

# 28.4 Functional Description

## 28.4.1 Flash Command Operations

Flash command operations are used to modify Flash memory contents or configure module resources for EEE operation.

The next sections describe:

- How to write the FCLKDIV register that is used to generate a time base (FCLK) derived from OSCCLK for Flash program and erase command operations
- The command write sequence used to set Flash command parameters and launch execution
- Valid Flash commands available for execution

### 28.4.1.1 Writing the FCLKDIV Register

Prior to issuing any Flash program or erase command after a reset, the user is required to write the FCLKDIV register to divide OSCCLK down to a target FCLK of 1 MHz. Table 28-9 shows recommended values for the FDIV field based on OSCCLK frequency.

**NOTE**

Programming or erasing the Flash memory cannot be performed if the bus clock runs at less than 1 MHz. Setting FDIV too high can destroy the Flash memory due to overstress. Setting FDIV too low can result in incomplete programming or erasure of the Flash memory cells.

When the FCLKDIV register is written, the FDIVLD bit is set automatically. If the FDIVLD bit is 0, the FCLKDIV register has not been written since the last reset. If the FCLKDIV register has not been written, any Flash program or erase command loaded during a command write sequence will not execute and the ACCERR bit in the FSTAT register will set.

### 28.4.1.2 Command Write Sequence

The Memory Controller will launch all valid Flash commands entered using a command write sequence.

Before launching a command, the ACCERR and FPVIOL bits in the FSTAT register must be clear (see Section 28.3.2.7) and the CCIF flag should be tested to determine the status of the current command write sequence. If CCIF is 0, the previous command write sequence is still active, a new command write sequence cannot be started, and all writes to the FCCOB register are ignored.

**CAUTION**

Writes to any Flash register must be avoided while a Flash command is active (CCIF=0) to prevent corruption of Flash register contents and Memory Controller behavior.

#### 28.4.1.2.1 Define FCCOB Contents

The FCCOB parameter fields must be loaded with all required parameters for the Flash command being executed. Access to the FCCOB parameter fields is controlled via the CCOBIX bits in the FCCOBIX register (see Section 28.3.2.3).

The contents of the FCCOB parameter fields are transferred to the Memory Controller when the user clears the CCIF command completion flag in the FSTAT register (writing 1 clears the CCIF to 0). The CCIF flag will remain clear until the Flash command has completed. Upon completion, the Memory Controller will return CCIF to 1 and the FCCOB register will be used to communicate any results. The flow for a generic command write sequence is shown in Figure 28-26.

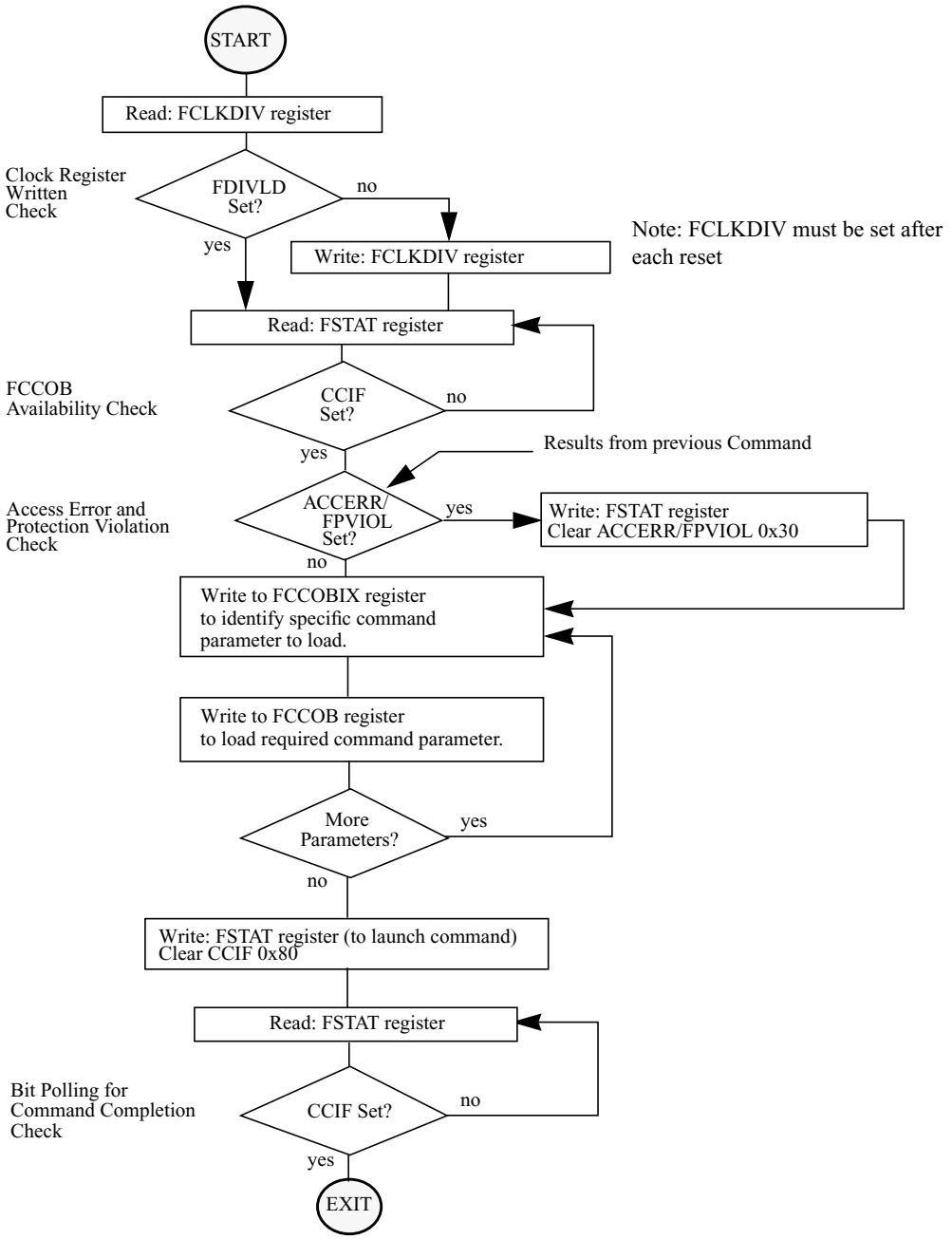



**Figure 28-26. Generic Flash Command Write Sequence Flowchart**

### 28.4.1.3 Valid Flash Module Commands

**Table 28-30. Flash Commands by Mode**

| FCMD | Command | Unsecured | | | | Secured | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | NS (1) | NX (2) | SS(3) | ST(4) | NS (5) | NX (6) | SS(7) | ST(8) |
| 0x01 | Erase Verify All Blocks | * | * | * | * | * | * | * | * |
| 0x02 | Erase Verify Block | * | * | * | * | * | * | * | * |
| 0x03 | Erase Verify P-Flash Section | * | * | * | * | * | | | |
| 0x04 | Read Once | * | * | * | * | * | | | |
| 0x05 | Load Data Field | * | * | * | * | * | | | |
| 0x06 | Program P-Flash | * | * | * | * | * | | | |
| 0x07 | Program Once | * | * | * | * | * | | | |
| 0x08 | Erase All Blocks | | | * | * | | | * | * |
| 0x09 | Erase P-Flash Block | * | * | * | * | * | | | |
| 0x0A | Erase P-Flash Sector | * | * | * | * | * | | | |
| 0x0B | Unsecure Flash | | | * | * | | | * | * |
| 0x0C | Verify Backdoor Access Key | * | | | | * | | | |
| 0x0D | Set User Margin Level | * | * | * | * | * | | | |
| 0x0E | Set Field Margin Level | | | * | * | | | | |
| 0x0F | Full Partition D-Flash | | | * | * | | | | |
| 0x10 | Erase Verify D-Flash Section | * | * | * | * | * | | | |
| 0x11 | Program D-Flash | * | * | * | * | * | | | |
| 0x12 | Erase D-Flash Sector | * | * | * | * | * | | | |
| 0x13 | Enable EEPROM Emulation | * | * | * | * | * | * | * | * |
| 0x14 | Disable EEPROM Emulation | * | * | * | * | * | * | * | * |
| 0x15 | EEPROM Emulation Query | * | * | * | * | * | * | * | * |
| 0x20 | Partition D-Flash | * | * | * | * | * | * | * | * |

1. Unsecured Normal Single Chip mode.
2. Unsecured Normal Expanded mode.
3. Unsecured Special Single Chip mode.
4. Unsecured Special Mode.
5. Secured Normal Single Chip mode.
6. Secured Normal Expanded mode.
7. Secured Special Single Chip mode.
8. Secured Special Mode.

### 28.4.1.4 P-Flash Commands

Table 28-31 summarizes the valid P-Flash commands along with the effects of the commands on the P-Flash block and other resources within the Flash module.

**Table 28-31. P-Flash Commands**

| FCMD | Command | Function on P-Flash Memory |
|---|---|---|
| 0x01 | Erase Verify All Blocks | Verify that all P-Flash (and D-Flash) blocks are erased. |
| 0x02 | Erase Verify Block | Verify that a P-Flash block is erased. |
| 0x03 | Erase Verify P-Flash Section | Verify that a given number of words starting at the address provided are erased. |
| 0x04 | Read Once | Read a dedicated 64 byte field in the nonvolatile information register in P-Flash block 0 that was previously programmed using the Program Once command. |
| 0x05 | Load Data Field | Load data for simultaneous multiple P-Flash block operations. |
| 0x06 | Program P-Flash | Program a phrase in a P-Flash block and any previously loaded phrases for any other P-Flash block (see Load Data Field command). |
| 0x07 | Program Once | Program a dedicated 64 byte field in the nonvolatile information register in P-Flash block 0 that is allowed to be programmed only once. |
| 0x08 | Erase All Blocks | Erase all P-Flash (and D-Flash) blocks.<br>An erase of all Flash blocks is only possible when the FPLDIS, FPHDIS, and FPOPEN bits in the FPROT register and the EPDIS and EPOPEN bits in the EPROT register are set prior to launching the command. |
| 0x09 | Erase P-Flash Block | Erase a single P-Flash block.<br>An erase of the full P-Flash block is only possible when FPLDIS, FPHDIS and FPOPEN bits in the FPROT register are set prior to launching the command. |
| 0x0A | Erase P-Flash Sector | Erase all bytes in a P-Flash sector. |
| 0x0B | Unsecure Flash | Supports a method of releasing MCU security by erasing all P-Flash (and D-Flash) blocks and verifying that all P-Flash (and D-Flash) blocks are erased. |
| 0x0C | Verify Backdoor Access Key | Supports a method of releasing MCU security by verifying a set of security keys. |
| 0x0D | Set User Margin Level | Specifies a user margin read level for all P-Flash blocks. |
| 0x0E | Set Field Margin Level | Specifies a field margin read level for all P-Flash blocks (special modes only). |

### 28.4.1.5 D-Flash and EEE Commands

Table 28-32 summarizes the valid D-Flash and EEE commands along with the effects of the commands on the D-Flash block and EEE operation.

**Table 28-32. D-Flash Commands**

| FCMD | Command | Function on D-Flash Memory |
|---|---|---|
| 0x01 | Erase Verify All Blocks | Verify that all D-Flash (and P-Flash) blocks are erased. |
| 0x02 | Erase Verify Block | Verify that the D-Flash block is erased. |

**Table 28-32. D-Flash Commands**

| FCMD | Command | Function on D-Flash Memory |
|---|---|---|
| 0x08 | Erase All Blocks | Erase all D-Flash (and P-Flash) blocks.<br>An erase of all Flash blocks is only possible when the FPLDIS, FPHDIS, and FPOPEN bits in the FPROT register and the EPDIS and EPOPEN bits in the EPROT register are set prior to launching the command. |
| 0x0B | Unsecure Flash | Supports a method of releasing MCU security by erasing all D-Flash (and P-Flash) blocks and verifying that all D-Flash (and P-Flash) blocks are erased. |
| 0x0D | Set User Margin Level | Specifies a user margin read level for the D-Flash block. |
| 0x0E | Set Field Margin Level | Specifies a field margin read level for the D-Flash block (special modes only). |
| 0x0F | Full Partition D-Flash | Erase the D-Flash block and partition an area of the D-Flash block for user access. |
| 0x10 | Erase Verify D-Flash Section | Verify that a given number of words starting at the address provided are erased. |
| 0x11 | Program D-Flash | Program up to four words in the D-Flash block. |
| 0x12 | Erase D-Flash Sector | Erase all bytes in a sector of the D-Flash block. |
| 0x13 | Enable EEPROM Emulation | Enable EEPROM emulation where writes to the buffer RAM EEE partition will be copied to the D-Flash EEE partition. |
| 0x14 | Disable EEPROM Emulation | Suspend all current erase and program activity related to EEPROM emulation but leave current EEE tags set. |
| 0x15 | EEPROM Emulation Query | Returns EEE partition and status variables. |
| 0x20 | Partition D-Flash | Partition an area of the D-Flash block for user access. |

## 28.4.2 Flash Command Description

This section provides details of all available Flash commands launched by a command write sequence. The ACCERR bit in the FSTAT register will be set during the command write sequence if any of the following illegal steps are performed, causing the command not to be processed by the Memory Controller:

- Starting any command write sequence that programs or erases Flash memory before initializing the FCLKDIV register
- Writing an invalid command as part of the command write sequence
- For additional possible errors, refer to the error handling table provided for each command

If a Flash block is read during execution of an algorithm (CCIF = 0) on that same block, the read operation will return invalid data. If the SFDIF or DFDIF flags were not previously set when the invalid read operation occurred, both the SFDIF and DFDIF flags will be set and the FECCR registers will be loaded with the global address used in the invalid read operation with the data and parity fields set to all 0.

If the ACCERR or FPVIOL bits are set in the FSTAT register, the user must clear these bits before starting any command write sequence (see Section 28.3.2.7).

**CAUTION**

A Flash word or phrase must be in the erased state before being programmed. Cumulative programming of bits within a Flash word or phrase is not allowed.

### 28.4.2.1 Erase Verify All Blocks Command

The Erase Verify All Blocks command will verify that all P-Flash and D-Flash blocks have been erased.

**Table 28-33. Erase Verify All Blocks Command FCCOB Requirements**

| CCOBIX[2:0] | FCCOB Parameters | |
|---|---|---|
| 000 | 0x01 | Not required |

Upon clearing CCIF to launch the Erase Verify All Blocks command, the Memory Controller will verify that the entire Flash memory space is erased. The CCIF flag will set after the Erase Verify All Blocks operation has completed.

**Table 28-34. Erase Verify All Blocks Command Error Handling**

| Register | Error Bit | Error Condition |
|---|---|---|
| FSTAT | ACCERR | Set if CCOBIX[2:0] != 000 at command launch |
| | | Set if a Load Data Field command sequence is currently active |
| | FPVIOL | None |
| | MGSTAT1 | Set if any errors have been encountered during the read(1) |
| | MGSTAT0 | Set if any non-correctable errors have been encountered during the read[1] |
| FERSTAT | EPVIOLIF | None |

1. As found in the memory map for FTM1024K5.

### 28.4.2.2 Erase Verify Block Command

The Erase Verify Block command allows the user to verify that an entire P-Flash or D-Flash block has been erased. The FCCOB upper global address bits determine which block must be verified.

**Table 28-35. Erase Verify Block Command FCCOB Requirements**

| CCOBIX[2:0] | FCCOB Parameters | |
|---|---|---|
| 000 | 0x02 | Global address [22:16] of the Flash block to be verified. |

Upon clearing CCIF to launch the Erase Verify Block command, the Memory Controller will verify that the selected P-Flash or D-Flash block is erased. The CCIF flag will set after the Erase Verify Block operation has completed.

**Table 28-36. Erase Verify Block Command Error Handling**

| Register | Error Bit | Error Condition |
|---|---|---|
| FSTAT | ACCERR | Set if CCOBIX[2:0] != 000 at command launch |
| | | Set if a Load Data Field command sequence is currently active |
| | | Set if an invalid global address [22:16] is supplied(1) |
| | FPVIOL | None |
| | MGSTAT1 | Set if any errors have been encountered during the read(2) |
| | MGSTAT0 | Set if any non-correctable errors have been encountered during the read2 |
| FERSTAT | EPVIOLIF | None |

1. As defined by the memory map for FTM1024K5.
2. As found in the memory map for FTM1024K5.

### 28.4.2.3 Erase Verify P-Flash Section Command

The Erase Verify P-Flash Section command will verify that a section of code in the P-Flash memory is erased. The Erase Verify P-Flash Section command defines the starting point of the code to be verified and the number of phrases. The section to be verified cannot cross a 256 Kbyte boundary in the P-Flash memory space.

**Table 28-37. Erase Verify P-Flash Section Command FCCOB Requirements**

| CCOBIX[2:0] | FCCOB Parameters | |
|---|---|---|
| 000 | 0x03 | Global address [22:16] of a P-Flash block |
| 001 | Global address [15:0] of the first phrase to be verified | |
| 010 | Number of phrases to be verified | |

Upon clearing CCIF to launch the Erase Verify P-Flash Section command, the Memory Controller will verify the selected section of Flash memory is erased. The CCIF flag will set after the Erase Verify P-Flash Section operation has completed.

**Table 28-38. Erase Verify P-Flash Section Command Error Handling**

| Register | Error Bit | Error Condition |
|---|---|---|
| FSTAT | ACCERR | Set if CCOBIX[2:0] != 010 at command launch |
| | | Set if a Load Data Field command sequence is currently active |
| | | Set if command not available in current mode (see Table 28-30) |
| | | Set if an invalid global address [22:0] is supplied(1) |
| | | Set if a misaligned phrase address is supplied (global address [2:0] != 000) |
| | | Set if the requested section crosses a 256 Kbyte boundary |
| | FPVIOL | None |
| | MGSTAT1 | Set if any errors have been encountered during the read(2) |
| | MGSTAT0 | Set if any non-correctable errors have been encountered during the read2 |
| FERSTAT | EPVIOLIF | None |

1. As defined by the memory map for FTM1024K5.
2. As found in the memory map for FTM1024K5.

### 28.4.2.4 Read Once Command

The Read Once command provides read access to a reserved 64 byte field (8 phrases) located in the nonvolatile information register of P-Flash block 0. The Read Once field is programmed using the Program Once command described in Section 28.4.2.7. The Read Once command must not be executed from the Flash block containing the Program Once reserved field to avoid code runaway.

**Table 28-39. Read Once Command FCCOB Requirements**

| CCOBIX[2:0] | FCCOB Parameters | |
|---|---|---|
| 000 | 0x04 | Not Required |
| 001 | Read Once phrase index (0x0000 - 0x0007) | |
| 010 | Read Once word 0 value | |
| 011 | Read Once word 1 value | |
| 100 | Read Once word 2 value | |
| 101 | Read Once word 3 value | |

Upon clearing CCIF to launch the Read Once command, a Read Once phrase is fetched and stored in the FCCOB indexed register. The CCIF flag will set after the Read Once operation has completed. Valid phrase index values for the Read Once command range from 0x0000 to 0x0007. During execution of the Read Once command, any attempt to read addresses within P-Flash block 0 will return invalid data.

**Table 28-40. Read Once Command Error Handling**

| Register | Error Bit | Error Condition |
|---|---|---|
| FSTAT | ACCERR | Set if CCOBIX[2:0] != 001 at command launch |
| | | Set if a Load Data Field command sequence is currently active |
| | | Set if command not available in current mode (see Table 28-30) |
| | | Set if an invalid phrase index is supplied |
| | FPVIOL | None |
| | MGSTAT1 | Set if any errors have been encountered during the read |
| | MGSTAT0 | Set if any non-correctable errors have been encountered during the read |
| FERSTAT | EPVIOLIF | None |

### 28.4.2.5 Load Data Field Command

The Load Data Field command is executed to provide FCCOB parameters for multiple P-Flash blocks for a future simultaneous program operation in the P-Flash memory space.

**Table 28-41. Load Data Field Command FCCOB Requirements**

| CCOBIX[2:0] | FCCOB Parameters | |
|---|---|---|
| 000 | 0x05 | Global address [22:16] to identify P-Flash block |
| 001 | Global address [15:0] of phrase location to be programmed(1) | |
| 010 | Word 0 | |
| 011 | Word 1 | |
| 100 | Word 2 | |
| 101 | Word 3 | |

1. Global address [2:0] must be 000

Upon clearing CCIF to launch the Load Data Field command, the FCCOB registers will be transferred to the Memory Controller and be programmed in the block specified at the global address given with a future Program P-Flash command launched on a P-Flash block. The CCIF flag will set after the Load Data Field operation has completed. Note that once a Load Data Field command sequence has been initiated, the Load Data Field command sequence will be cancelled if any command other than Load Data Field or the future Program P-Flash is launched. Similarly, if an error occurs after launching a Load Data Field or Program P-Flash command, the associated Load Data Field command sequence will be cancelled.

**Table 28-42. Load Data Field Command Error Handling**

| Register | Error Bit | Error Condition |
|---|---|---|
| FSTAT | ACCERR | Set if CCOBIX[2:0] != 101 at command launch |
| | | Set if command not available in current mode (see Table 28-30) |
| | | Set if an invalid global address [22:0] is supplied(1) |
| | | Set if a misaligned phrase address is supplied (global address [2:0] != 000) |
| | | Set if a Load Data Field command sequence is currently active and the selected block has previously been selected in the same command sequence |
| | | Set if a Load Data Field command sequence is currently active and global address [17:0] does not match that previously supplied in the same command sequence |
| | FPVIOL | Set if the global address [22:0] points to a protected area |
| | MGSTAT1 | None |
| | MGSTAT0 | None |
| FERSTAT | EPVIOLIF | None |

1. As defined by the memory map for FTM1024K5.

### 28.4.2.6 Program P-Flash Command

The Program P-Flash operation will program a previously erased phrase in the P-Flash memory using an embedded algorithm.

**CAUTION**

A P-Flash phrase must be in the erased state before being programmed. Cumulative programming of bits within a Flash phrase is not allowed.

**Table 28-43. Program P-Flash Command FCCOB Requirements**

| CCOBIX[2:0] | FCCOB Parameters | |
|---|---|---|
| 000 | 0x06 | Global address [22:16] to identify P-Flash block |
| 001 | Global address [15:0] of phrase location to be programmed(1) | |
| 010 | Word 0 program value | |
| 011 | Word 1 program value | |
| 100 | Word 2 program value | |
| 101 | Word 3 program value | |

1. Global address [2:0] must be 000

Upon clearing CCIF to launch the Program P-Flash command, the Memory Controller will program the data words to the supplied global address and will then proceed to verify the data words read back as expected. The CCIF flag will set after the Program P-Flash operation has completed.

**Table 28-44. Program P-Flash Command Error Handling**

| Register | Error Bit | Error Condition |
|---|---|---|
| FSTAT | ACCERR | Set if CCOBIX[2:0] != 101 at command launch |
| | | Set if command not available in current mode (see Table 28-30) |
| | | Set if an invalid global address [22:0] is supplied(1) |
| | | Set if a misaligned phrase address is supplied (global address [2:0] != 000) |
| | | Set if a Load Data Field command sequence is currently active and the selected block has previously been selected in the same command sequence |
| | | Set if a Load Data Field command sequence is currently active and global address [17:0] does not match that previously supplied in the same command sequence |
| | FPVIOL | Set if the global address [22:0] points to a protected area |
| | MGSTAT1 | Set if any errors have been encountered during the verify operation |
| | MGSTAT0 | Set if any non-correctable errors have been encountered during the verify operation |
| FERSTAT | EPVIOLIF | None |

1. As defined by the memory map for FTM1024K5.

### 28.4.2.7 Program Once Command

The Program Once command restricts programming to a reserved 64 byte field (8 phrases) in the nonvolatile information register located in P-Flash block 0. The Program Once reserved field can be read using the Read Once command as described in Section 28.4.2.4. The Program Once command must only be issued once since the nonvolatile information register in P-Flash block 0 cannot be erased. The Program Once command must not be executed from the Flash block containing the Program Once reserved field to avoid code runaway.

**Table 28-45. Program Once Command FCCOB Requirements**

| CCOBIX[2:0] | FCCOB Parameters | |
|---|---|---|
| 000 | 0x07 | Not Required |
| 001 | Program Once phrase index (0x0000 - 0x0007) | |
| 010 | Program Once word 0 value | |
| 011 | Program Once word 1 value | |
| 100 | Program Once word 2 value | |
| 101 | Program Once word 3 value | |

Upon clearing CCIF to launch the Program Once command, the Memory Controller first verifies that the selected phrase is erased. If erased, then the selected phrase will be programmed and then verified with read back. The CCIF flag will remain clear, setting only after the Program Once operation has completed.

The reserved nonvolatile information register accessed by the Program Once command cannot be erased and any attempt to program one of these phrases a second time will not be allowed. Valid phrase index

values for the Program Once command range from 0x0000 to 0x0007. During execution of the Program Once command, any attempt to read addresses within P-Flash block 0 will return invalid data.

**Table 28-46. Program Once Command Error Handling**

| Register | Error Bit | Error Condition |
|---|---|---|
| FSTAT | ACCERR | Set if CCOBIX[2:0] != 101 at command launch |
| | | Set if a Load Data Field command sequence is currently active |
| | | Set if command not available in current mode (see Table 28-30) |
| | | Set if an invalid phrase index is supplied |
| | | Set if the requested phrase has already been programmed(1) |
| | FPVIOL | None |
| | MGSTAT1 | Set if any errors have been encountered during the verify operation |
| | MGSTAT0 | Set if any non-correctable errors have been encountered during the verify operation |
| FERSTAT | EPVIOLIF | None |

1. If a Program Once phrase is initially programmed to 0xFFFF_FFFF_FFFF_FFFF, the Program Once command will be allowed to execute again on that same phrase.

### 28.4.2.8 Erase All Blocks Command

The Erase All Blocks operation will erase the entire P-Flash and D-Flash memory space including the EEE nonvolatile information register.

**Table 28-47. Erase All Blocks Command FCCOB Requirements**

| CCOBIX[2:0] | FCCOB Parameters | |
|---|---|---|
| 000 | 0x08 | Not required |

Upon clearing CCIF to launch the Erase All Blocks command, the Memory Controller will erase the entire Flash memory space and verify that it is erased. If the Memory Controller verifies that the entire Flash memory space was properly erased, security will be released. During the execution of this command (CCIF=0) the user must not write to any Flash module register. The CCIF flag will set after the Erase All Blocks operation has completed.

**Table 28-48. Erase All Blocks Command Error Handling**

| Register | Error Bit | Error Condition |
|---|---|---|
| FSTAT | ACCERR | Set if CCOBIX[2:0] != 000 at command launch |
| | | Set if a Load Data Field command sequence is currently active |
| | | Set if command not available in current mode (see Table 28-30) |
| | FPVIOL | Set if any area of the P-Flash memory is protected |
| | MGSTAT1 | Set if any errors have been encountered during the verify operation(1) |
| | MGSTAT0 | Set if any non-correctable errors have been encountered during the verify operation[1] |
| FERSTAT | EPVIOLIF | Set if any area of the buffer RAM EEE partition is protected |

1. As found in the memory map for FTM1024K5.

### 28.4.2.9 Erase P-Flash Block Command

The Erase P-Flash Block operation will erase all addresses in a P-Flash block.

**Table 28-49. Erase P-Flash Block Command FCCOB Requirements**

| CCOBIX[2:0] | FCCOB Parameters | |
|---|---|---|
| 000 | 0x09 | Global address [22:16] to identify P-Flash block |
| 001 | Global address [15:0] in P-Flash block to be erased | |

Upon clearing CCIF to launch the Erase P-Flash Block command, the Memory Controller will erase the selected P-Flash block and verify that it is erased. The CCIF flag will set after the Erase P-Flash Block operation has completed.

**Table 28-50. Erase P-Flash Block Command Error Handling**

| Register | Error Bit | Error Condition |
|---|---|---|
| FSTAT | ACCERR | Set if CCOBIX[2:0] != 001 at command launch |
| | | Set if a Load Data Field command sequence is currently active |
| | | Set if command not available in current mode (see Table 28-30) |
| | | Set if an invalid global address [22:16] is supplied(1) |
| | FPVIOL | Set if an area of the selected P-Flash block is protected |
| | MGSTAT1 | Set if any errors have been encountered during the verify operation(2) |
| | MGSTAT0 | Set if any non-correctable errors have been encountered during the verify operation[2] |
| FERSTAT | EPVIOLIF | None |

1. As defined by the memory map for FTM1024K5.
2. As found in the memory map for FTM1024K5.

### 28.4.2.10 Erase P-Flash Sector Command

The Erase P-Flash Sector operation will erase all addresses in a P-Flash sector.

**Table 28-51. Erase P-Flash Sector Command FCCOB Requirements**

| CCOBIX[2:0] | FCCOB Parameters | |
|---|---|---|
| 000 | 0x0A | Global address [22:16] to identify P-Flash block to be erased |
| 001 | Global address [15:0] anywhere within the sector to be erased. Refer to Section 28.1.2.1 for the P-Flash sector size. | |

Upon clearing CCIF to launch the Erase P-Flash Sector command, the Memory Controller will erase the selected Flash sector and then verify that it is erased. The CCIF flag will be set after the Erase P-Flash Sector operation has completed.

**Table 28-52. Erase P-Flash Sector Command Error Handling**

| Register | Error Bit | Error Condition |
|---|---|---|
| FSTAT | ACCERR | Set if CCOBIX[2:0] != 001 at command launch |
| | | Set if a Load Data Field command sequence is currently active |
| | | Set if command not available in current mode (see Table 28-30) |
| | | Set if an invalid global address [22:16] is supplied(1) |
| | | Set if a misaligned phrase address is supplied (global address [2:0] != 000) |
| | FPVIOL | Set if the selected P-Flash sector is protected |
| | MGSTAT1 | Set if any errors have been encountered during the verify operation |
| | MGSTAT0 | Set if any non-correctable errors have been encountered during the verify operation |
| FERSTAT | EPVIOLIF | None |

1. As defined by the memory map for FTM1024K5.

### 28.4.2.11 Unsecure Flash Command

The Unsecure Flash command will erase the entire P-Flash and D-Flash memory space and, if the erase is successful, will release security.

**Table 28-53. Unsecure Flash Command FCCOB Requirements**

| CCOBIX[2:0] | FCCOB Parameters | |
|---|---|---|
| 000 | 0x0B | Not required |

Upon clearing CCIF to launch the Unsecure Flash command, the Memory Controller will erase the entire P-Flash and D-Flash memory space and verify that it is erased. If the Memory Controller verifies that the entire Flash memory space was properly erased, security will be released. If the erase verify is not successful, the Unsecure Flash operation sets MGSTAT1 and terminates without changing the security

state. During the execution of this command (CCIF=0) the user must not write to any Flash module register. The CCIF flag is set after the Unsecure Flash operation has completed.

**Table 28-54. Unsecure Flash Command Error Handling**

| Register | Error Bit | Error Condition |
|---|---|---|
| FSTAT | ACCERR | Set if CCOBIX[2:0] != 000 at command launch |
| | | Set if a Load Data Field command sequence is currently active |
| | | Set if command not available in current mode (see Table 28-30) |
| | FPVIOL | Set if any area of the P-Flash memory is protected |
| | MGSTAT1 | Set if any errors have been encountered during the verify operation(1) |
| | MGSTAT0 | Set if any non-correctable errors have been encountered during the verify operation[1] |
| FERSTAT | EPVIOLIF | Set if any area of the buffer RAM EEE partition is protected |

1. As found in the memory map for FTM1024K5.

### 28.4.2.12 Verify Backdoor Access Key Command

The Verify Backdoor Access Key command will only execute if it is enabled by the KEYEN bits in the FSEC register (see Table 28-11). The Verify Backdoor Access Key command releases security if user-supplied keys match those stored in the Flash security bytes of the Flash configuration field (see Table 28-3). The Verify Backdoor Access Key command must not be executed from the Flash block containing the backdoor comparison key to avoid code runaway.

**Table 28-55. Verify Backdoor Access Key Command FCCOB Requirements**

| CCOBIX[2:0] | FCCOB Parameters | |
|---|---|---|
| 000 | 0x0C | Not required |
| 001 | Key 0 | |
| 010 | Key 1 | |
| 011 | Key 2 | |
| 100 | Key 3 | |

Upon clearing CCIF to launch the Verify Backdoor Access Key command, the Memory Controller will check the FSEC KEYEN bits to verify that this command is enabled. If not enabled, the Memory Controller sets the ACCERR bit in the FSTAT register and terminates. If the command is enabled, the Memory Controller compares the key provided in FCCOB to the backdoor comparison key in the Flash configuration field with Key 0 compared to 0x7F_FF00, etc. If the backdoor keys match, security will be released. If the backdoor keys do not match, security is not released and all future attempts to execute the Verify Backdoor Access Key command are aborted (set ACCERR) until a reset occurs. The CCIF flag is set after the Verify Backdoor Access Key operation has completed.

**Table 28-56. Verify Backdoor Access Key Command Error Handling**

| Register | Error Bit | Error Condition |
|---|---|---|
| FSTAT | ACCERR | Set if CCOBIX[2:0] != 100 at command launch |
| | | Set if a Load Data Field command sequence is currently active |
| | | Set if an incorrect backdoor key is supplied |
| | | Set if backdoor key access has not been enabled (KEYEN[1:0] != 10, see Section 28.3.2.2) |
| | | Set if the backdoor key has mismatched since the last reset |
| | FPVIOL | None |
| | MGSTAT1 | None |
| | MGSTAT0 | None |
| FERSTAT | EPVIOLIF | None |

### 28.4.2.13 Set User Margin Level Command

The Set User Margin Level command causes the Memory Controller to set the margin level for future read operations of a specific P-Flash or D-Flash block.

**Table 28-57. Set User Margin Level Command FCCOB Requirements**

| CCOBIX[2:0] | FCCOB Parameters | |
|---|---|---|
| 000 | 0x0D | Global address [22:16] to identify the Flash block |
| 001 | Margin level setting | |

Upon clearing CCIF to launch the Set User Margin Level command, the Memory Controller will set the user margin level for the targeted block and then set the CCIF flag.

Valid margin level settings for the Set User Margin Level command are defined in Table 28-58.

**Table 28-58. Valid Set User Margin Level Settings**

| CCOB (CCOBIX=001) | Level Description |
|---|---|
| 0x0000 | Return to Normal Level |
| 0x0001 | User Margin-1 Level(1) |
| 0x0002 | User Margin-0 Level(2) |

1. Read margin to the erased state
2. Read margin to the programmed state

**Table 28-59. Set User Margin Level Command Error Handling**

| Register | Error Bit | Error Condition |
|---|---|---|
| FSTAT | ACCERR | Set if CCOBIX[2:0] != 001 at command launch |
| | | Set if a Load Data Field command sequence is currently active |
| | | Set if command not available in current mode (see Table 28-30) |
| | | Set if an invalid global address [22:16] is supplied(1) |
| | | Set if an invalid margin level setting is supplied |
| | FPVIOL | None |
| | MGSTAT1 | None |
| | MGSTAT0 | None |
| FERSTAT | EPVIOLIF | None |

1. As defined by the memory map for FTM1024K5.

**NOTE**

User margin levels can be used to check that Flash memory contents have adequate margin for normal level read operations. If unexpected results are encountered when checking Flash memory contents at user margin levels, a potential loss of information has been detected.

### 28.4.2.14 Set Field Margin Level Command

The Set Field Margin Level command, valid in special modes only, causes the Memory Controller to set the margin level specified for future read operations of a specific P-Flash or D-Flash block.

**Table 28-60. Set Field Margin Level Command FCCOB Requirements**

| CCOBIX[2:0] | FCCOB Parameters | |
|---|---|---|
| 000 | 0x0E | Global address [22:16] to identify the Flash block |
| 001 | Margin level setting | |

Upon clearing CCIF to launch the Set Field Margin Level command, the Memory Controller will set the field margin level for the targeted block and then set the CCIF flag.

Valid margin level settings for the Set Field Margin Level command are defined in Table 28-61.

**Table 28-61. Valid Set Field Margin Level Settings**

| CCOB (CCOBIX=001) | Level Description |
|---|---|
| 0x0000 | Return to Normal Level |
| 0x0001 | User Margin-1 Level(1) |

**Table 28-61. Valid Set Field Margin Level Settings**

| CCOB (CCOBIX=001) | Level Description |
|---|---|
| 0x0002 | User Margin-0 Level(2) |
| 0x0003 | Field Margin-1 Level1 |
| 0x0004 | Field Margin-0 Level2 |

1. Read margin to the erased state
2. Read margin to the programmed state

**Table 28-62. Set Field Margin Level Command Error Handling**

| Register | Error Bit | Error Condition |
|---|---|---|
| FSTAT | ACCERR | Set if CCOBIX[2:0] != 001 at command launch |
| | | Set if a Load Data Field command sequence is currently active |
| | | Set if command not available in current mode (see Table 28-30) |
| | | Set if an invalid global address [22:16] is supplied(1) |
| | | Set if an invalid margin level setting is supplied |
| | FPVIOL | None |
| | MGSTAT1 | None |
| | MGSTAT0 | None |
| FERSTAT | EPVIOLIF | None |

1. As defined by the memory map for FTM1024K5.

**CAUTION**

Field margin levels must only be used during verify of the initial factory programming.

**NOTE**

Field margin levels can be used to check that Flash memory contents have adequate margin for data retention at the normal level setting. If unexpected results are encountered when checking Flash memory contents at field margin levels, the Flash memory contents should be erased and reprogrammed.

### 28.4.2.15 Full Partition D-Flash Command

The Full Partition D-Flash command allows the user to allocate sectors within the D-Flash block for applications and a partition within the buffer RAM for EEPROM access. The D-Flash block consists of 128 sectors with 256 bytes per sector.

**Table 28-63. Full Partition D-Flash Command FCCOB Requirements**

| CCOBIX[2:0] | FCCOB Parameters | |
|---|---|---|
| 000 | 0x0F | Not required |
| 001 | Number of 256 byte sectors for the D-Flash user partition (DFPART) | |
| 010 | Number of 256 byte sectors for buffer RAM EEE partition (ERPART) | |

Upon clearing CCIF to launch the Full Partition D-Flash command, the following actions are taken to define a partition within the D-Flash block for direct access (DFPART) and a partition within the buffer RAM for EEE use (ERPART):

- Validate the DFPART and ERPART values provided:
  - DFPART <= 128 (maximum number of 256 byte sectors in D-Flash block)
  - ERPART <= 16 (maximum number of 256 byte sectors in buffer RAM)
  - If ERPART > 0, 128 - DFPART >= 12 (minimum number of 256 byte sectors in the D-Flash block required to support EEE)
  - If ERPART > 0, ((128-DFPART)/ERPART) >= 8 (minimum ratio of D-Flash EEE space to buffer RAM EEE space to support EEE)
- Erase the D-Flash block and the EEE nonvolatile information register
- Program DFPART to the EEE nonvolatile information register at global address 0x12_0000 (see Table 28-7)
- Program a duplicate DFPART to the EEE nonvolatile information register at global address 0x12_0002 (see Table 28-7)
- Program ERPART to the EEE nonvolatile information register at global address 0x12_0004 (see Table 28-7)
- Program a duplicate ERPART to the EEE nonvolatile information register at global address 0x12_0006 (see Table 28-7)

The D-Flash user partition will start at global address 0x10_0000. The buffer RAM EEE partition will end at global address 0x13_FFFF. After the Full Partition D-Flash operation has completed, the CCIF flag will set.

Running the Full Partition D-Flash command a second time will result in the previous partition values and the entire D-Flash memory being erased. The data value written corresponds to the number of 256 byte sectors allocated for either direct D-Flash access (DFPART) or buffer RAM EEE access (ERPART).

**Table 28-64. Full Partition D-Flash Command Error Handling**

| Register | Error Bit | Error Condition |
|---|---|---|
| FSTAT | ACCERR | Set if CCOBIX[2:0] != 010 at command launch |
| | | Set if a Load Data Field command sequence is currently active |
| | | Set if command not available in current mode (see Table 28-30) |
| | | Set if an invalid DFPART or ERPART selection is supplied |
| | FPVIOL | None |
| | MGSTAT1 | Set if any errors have been encountered during the read |
| | MGSTAT0 | Set if any non-correctable errors have been encountered during the read |
| FERSTAT | EPVIOLIF | None |

### 28.4.2.16 Erase Verify D-Flash Section Command

The Erase Verify D-Flash Section command will verify that a section of code in the D-Flash user partition is erased. The Erase Verify D-Flash Section command defines the starting point of the data to be verified and the number of words.

**Table 28-65. Erase Verify D-Flash Section Command FCCOB Requirements**

| CCOBIX[2:0] | FCCOB Parameters | |
|---|---|---|
| 000 | 0x10 | Global address [22:16] to identify the D-Flash block |
| 001 | Global address [15:0] of the first word to be verified | |
| 010 | Number of words to be verified | |

Upon clearing CCIF to launch the Erase Verify D-Flash Section command, the Memory Controller will verify the selected section of D-Flash memory is erased. The CCIF flag will set after the Erase Verify D-Flash Section operation has completed.

**Table 28-66. Erase Verify D-Flash Section Command Error Handling**

| Register | Error Bit | Error Condition |
|---|---|---|
| FSTAT | ACCERR | Set if CCOBIX[2:0] != 010 at command launch |
| | | Set if a Load Data Field command sequence is currently active |
| | | Set if command not available in current mode (see Table 28-30) |
| | | Set if an invalid global address [22:0] is supplied |
| | | Set if a misaligned word address is supplied (global address [0] != 0) |
| | | Set if the global address [22:0] points to an area of the D-Flash EEE partition |
| | | Set if the requested section breaches the end of the D-Flash block or goes into the D-Flash EEE partition |
| | FPVIOL | None |
| | MGSTAT1 | Set if any errors have been encountered during the read |
| | MGSTAT0 | Set if any non-correctable errors have been encountered during the read |
| FERSTAT | EPVIOLIF | None |

### 28.4.2.17 Program D-Flash Command

The Program D-Flash operation programs one to four previously erased words in the D-Flash user partition. The Program D-Flash operation will confirm that the targeted location(s) were successfully programmed upon completion.

**CAUTION**

A Flash word must be in the erased state before being programmed. Cumulative programming of bits within a Flash word is not allowed.

**Table 28-67. Program D-Flash Command FCCOB Requirements**

| CCOBIX[2:0] | FCCOB Parameters | |
|---|---|---|
| 000 | 0x11 | Global address [22:16] to identify the D-Flash block |
| 001 | Global address [15:0] of word to be programmed | |
| 010 | Word 0 program value | |
| 011 | Word 1 program value, if desired | |
| 100 | Word 2 program value, if desired | |
| 101 | Word 3 program value, if desired | |

Upon clearing CCIF to launch the Program D-Flash command, the user-supplied words will be transferred to the Memory Controller and be programmed. The CCOBIX index value at Program D-Flash command launch determines how many words will be programmed in the D-Flash block. No protection checks are made in the Program D-Flash operation on the D-Flash block, only access error checks. The CCIF flag is set when the operation has completed.

**Table 28-68. Program D-Flash Command Error Handling**

| Register | Error Bit | Error Condition |
|---|---|---|
| FSTAT | ACCERR | Set if CCOBIX[2:0] < 010 at command launch |
| | | Set if CCOBIX[2:0] > 101 at command launch |
| | | Set if a Load Data Field command sequence is currently active |
| | | Set if command not available in current mode (see Table 28-30) |
| | | Set if an invalid global address [22:0] is supplied |
| | | Set if a misaligned word address is supplied (global address [0] != 0) |
| | | Set if the global address [22:0] points to an area in the D-Flash EEE partition |
| | | Set if the requested group of words breaches the end of the D-Flash block or goes into the D-Flash EEE partition |
| | FPVIOL | None |
| | MGSTAT1 | Set if any errors have been encountered during the verify operation |
| | MGSTAT0 | Set if any non-correctable errors have been encountered during the verify operation |
| FERSTAT | EPVIOLIF | None |

### 28.4.2.18 Erase D-Flash Sector Command

The Erase D-Flash Sector operation will erase all addresses in a sector of the D-Flash user partition.

**Table 28-69. Erase D-Flash Sector Command FCCOB Requirements**

| CCOBIX[2:0] | FCCOB Parameters | |
|---|---|---|
| 000 | 0x12 | Global address [22:16] to identify D-Flash block |
| 001 | Global address [15:0] anywhere within the sector to be erased. See Section 28.1.2.2 for D-Flash sector size. | |

Upon clearing CCIF to launch the Erase D-Flash Sector command, the Memory Controller will erase the selected Flash sector and verify that it is erased. The CCIF flag will set after the Erase D-Flash Sector operation has completed.

**Table 28-70. Erase D-Flash Sector Command Error Handling**

| Register | Error Bit | Error Condition |
|---|---|---|
| FSTAT | ACCERR | Set if CCOBIX[2:0] != 001 at command launch |
| | | Set if a Load Data Field command sequence is currently active |
| | | Set if command not available in current mode (see Table 28-30) |
| | | Set if an invalid global address [22:0] is supplied |
| | | Set if a misaligned word address is supplied (global address [0] != 0) |
| | | Set if the global address [22:0] points to the D-Flash EEE partition |
| | FPVIOL | None |
| | MGSTAT1 | Set if any errors have been encountered during the verify operation |
| | MGSTAT0 | Set if any non-correctable errors have been encountered during the verify operation |
| FERSTAT | EPVIOLIF | None |

### 28.4.2.19 Enable EEPROM Emulation Command

The Enable EEPROM Emulation command causes the Memory Controller to enable EEE activity. EEE activity is disabled after any reset.

**Table 28-71. Enable EEPROM Emulation Command FCCOB Requirements**

| CCOBIX[2:0] | FCCOB Parameters | |
|---|---|---|
| 000 | 0x13 | Not required |

Upon clearing CCIF to launch the Enable EEPROM Emulation command, the CCIF flag will set after the Memory Controller enables EEE operations using the contents of the EEE tag RAM and tag counter. The Full Partition D-Flash or the Partition D-Flash command must be run prior to launching the Enable EEPROM Emulation command.

**Table 28-72. Enable EEPROM Emulation Command Error Handling**

| Register | Error Bit | Error Condition |
|---|---|---|
| FSTAT | ACCERR | Set if CCOBIX[2:0] != 000 at command launch |
| | | Set if a Load Data Field command sequence is currently active |
| | | Set if Full Partition D-Flash or Partition D-Flash command not previously run |
| | FPVIOL | None |
| | MGSTAT1 | None |
| | MGSTAT0 | None |
| FERSTAT | EPVIOLIF | None |

### 28.4.2.20 Disable EEPROM Emulation Command

The Disable EEPROM Emulation command causes the Memory Controller to suspend current EEE activity.

**Table 28-73. Disable EEPROM Emulation Command FCCOB Requirements**

| CCOBIX[2:0] | FCCOB Parameters | |
|---|---|---|
| 000 | 0x14 | Not required |

Upon clearing CCIF to launch the Disable EEPROM Emulation command, the Memory Controller will halt EEE operations at the next convenient point without clearing the EEE tag RAM or tag counter before setting the CCIF flag.

**Table 28-74. Disable EEPROM Emulation Command Error Handling**

| Register | Error Bit | Error Condition |
|---|---|---|
| FSTAT | ACCERR | Set if CCOBIX[2:0] != 000 at command launch |
| | | Set if a Load Data Field command sequence is currently active |
| | | Set if Full Partition D-Flash or Partition D-Flash command not previously run |
| | FPVIOL | None |
| | MGSTAT1 | None |
| | MGSTAT0 | None |
| FERSTAT | EPVIOLIF | None |

### 28.4.2.21 EEPROM Emulation Query Command

The EEPROM Emulation Query command returns EEE partition and status variables.

**Table 28-75. EEPROM Emulation Query Command FCCOB Requirements**

| CCOBIX[2:0] | FCCOB Parameters | |
|---|---|---|
| 000 | 0x15 | Not required |
| 001 | Return DFPART | |
| 010 | Return ERPART | |
| 011 | Return ECOUNT(1) | |
| 100 | Return Dead Sector Count | Return Ready Sector Count |

1. Indicates sector erase count

Upon clearing CCIF to launch the EEPROM Emulation Query command, the CCIF flag will set after the EEE partition and status variables are stored in the FCCOBIX register.If the Emulation Query command is executed prior to partitioning (Partition D-Flash Command Section 28.4.2.15), the following reset values are returned: DFPART = 0x_FFFF, ERPART = 0x_FFFF, ECOUNT = 0x_FFFF, Dead Sector Count = 0x_00, Ready Sector Count = 0x_00.

**Table 28-76. EEPROM Emulation Query Command Error Handling**

| Register | Error Bit | Error Condition |
|---|---|---|
| FSTAT | ACCERR | Set if CCOBIX[2:0] != 000 at command launch |
| | | Set if a Load Data Field command sequence is currently active |
| | | Set if command not available in current mode (see Table 28-30) |
| | FPVIOL | None |
| | MGSTAT1 | None |
| | MGSTAT0 | None |
| FERSTAT | EPVIOLIF | None |

### 28.4.2.22 Partition D-Flash Command

The Partition D-Flash command allows the user to allocate sectors within the D-Flash block for applications and a partition within the buffer RAM for EEPROM access. The D-Flash block consists of 128 sectors with 256 bytes per sector. The Erase All Blocks command must be run prior to launching the Partition D-Flash command.

**Table 28-77. Partition D-Flash Command FCCOB Requirements**

| CCOBIX[2:0] | FCCOB Parameters | |
|---|---|---|
| 000 | 0x20 | Not required |
| 001 | Number of 256 byte sectors for the D-Flash user partition (DFPART) | |
| 010 | Number of 256 byte sectors for buffer RAM EEE partition (ERPART) | |

Upon clearing CCIF to launch the Partition D-Flash command, the following actions are taken to define a partition within the D-Flash block for direct access (DFPART) and a partition within the buffer RAM for EEE use (ERPART):

- Validate the DFPART and ERPART values provided:
  - DFPART <= 128 (maximum number of 256 byte sectors in D-Flash block)
  - ERPART <= 16 (maximum number of 256 byte sectors in buffer RAM)
  - If ERPART > 0, 128 - DFPART >= 12 (minimum number of 256 byte sectors in the D-Flash block required to support EEE)
  - If ERPART > 0, ((128-DFPART)/ERPART) >= 8 (minimum ratio of D-Flash EEE space to buffer RAM EEE space to support EEE)
- Erase verify the D-Flash block and the EEE nonvolatile information register
- Program DFPART to the EEE nonvolatile information register at global address 0x12_0000 (see Table 28-7)

- Program a duplicate DFPART to the EEE nonvolatile information register at global address 0x12_0002 (see Table 28-7)
- Program ERPART to the EEE nonvolatile information register at global address 0x12_0004 (see Table 28-7)
- Program a duplicate ERPART to the EEE nonvolatile information register at global address 0x12_0006 (see Table 28-7)

The D-Flash user partition will start at global address 0x10_0000. The buffer RAM EEE partition will end at global address 0x13_FFFF. After the Partition D-Flash operation has completed, the CCIF flag will set.

Running the Partition D-Flash command a second time will result in the ACCERR bit within the FSTAT register being set. The data value written corresponds to the number of 256 byte sectors allocated for either direct D-Flash access (DFPART) or buffer RAM EEE access (ERPART).

**Table 28-78. Partition D-Flash Command Error Handling**

| Register | Error Bit | Error Condition |
|---|---|---|
| FSTAT | ACCERR | Set if CCOBIX[2:0] != 010 at command launch |
| | | Set if a Load Data Field command sequence is currently active |
| | | Set if command not available in current mode (see Table 28-30) |
| | | Set if partitions have already been defined |
| | | Set if an invalid DFPART or ERPART selection is supplied |
| | FPVIOL | None |
| | MGSTAT1 | Set if any errors have been encountered during the read |
| | MGSTAT0 | Set if any non-correctable errors have been encountered during the read |
| FERSTAT | EPVIOLIF | None |

## 28.4.3 Interrupts

The Flash module can generate an interrupt when a Flash command operation has completed or when a Flash command operation has detected an EEE error or an ECC fault.

**Table 28-79. Flash Interrupt Sources**

| Interrupt Source | Interrupt Flag | Local Enable | Global (CCR) Mask |
|---|---|---|---|
| Flash Command Complete | CCIF (FSTAT register) | CCIE (FCNFG register) | I Bit |
| Flash EEE Erase Error | ERSERIF (FERSTAT register) | ERSERIE (FERCNFG register) | I Bit |
| Flash EEE Program Error | PGMERIF (FERSTAT register) | PGMERIE (FERCNFG register) | I Bit |
| Flash EEE Protection Violation | EPVIOLIF (FERSTAT register) | EPVIOLIE (FERCNFG register) | I Bit |
| Flash EEE Error Type 1 Violation | ERSVIF1 (FERSTAT register) | ERSVIE1 (FERCNFG register) | I Bit |
| Flash EEE Error Type 0 Violation | ERSVIF0 (FERSTAT register) | ERSVIE0 (FERCNFG register) | I Bit |
| ECC Double Bit Fault on Flash Read | DFDIF (FERSTAT register) | DFDIE (FERCNFG register) | I Bit |
| ECC Single Bit Fault on Flash Read | SFDIF (FERSTAT register) | SFDIE (FERCNFG register) | I Bit |

**NOTE**

Vector addresses and their relative interrupt priority are determined at the MCU level.

### 28.4.3.1 Description of Flash Interrupt Operation

The Flash module uses the CCIF flag in combination with the CCIE interrupt enable bit to generate the Flash command interrupt request. The Flash module uses the ERSEIF, PGMEIF, EPVIOLIF, ERSVIF1, ERSVIF0, DFDIF and SFDIF flags in combination with the ERSEIE, PGMEIE, EPVIOLIE, ERSVIE1, ERSVIE0, DFDIE and SFDIE interrupt enable bits to generate the Flash error interrupt request. For a detailed description of the register bits involved, refer to Section 28.3.2.5, “Flash Configuration Register (FCNFG)”, Section 28.3.2.6, “Flash Error Configuration Register (FERCNFG)”, Section 28.3.2.7, “Flash Status Register (FSTAT)”, and Section 28.3.2.8, “Flash Error Status Register (FERSTAT)”.

The logic used for generating the Flash module interrupts is shown in Figure 28-27.

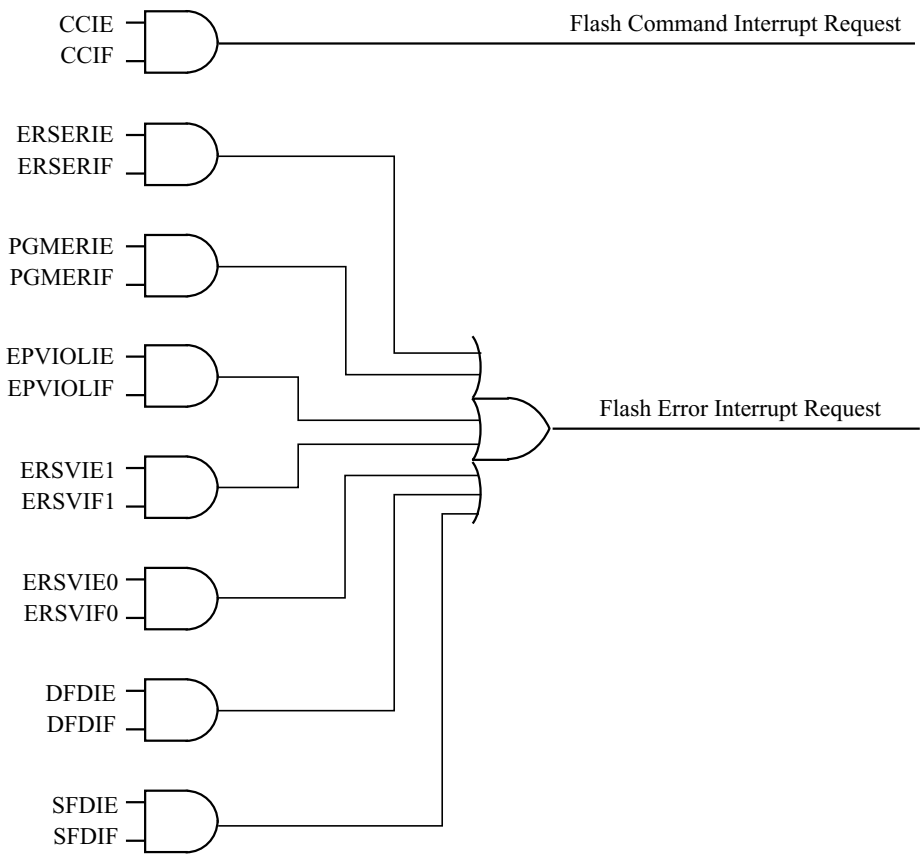



**Figure 28-27. Flash Module Interrupts Implementation**

### 28.4.4 Wait Mode

The Flash module is not affected if the MCU enters wait mode. The Flash module can recover the MCU from wait via the CCIF interrupt (see Section 28.4.3, “Interrupts”).

### 28.4.5 Stop Mode

If a Flash command is active (CCIF = 0) or an EE-Emulation operation is pending when the MCU requests stop mode, the current Flash operation will be completed before the CPU is allowed to enter stop mode.

## 28.5 Security

The Flash module provides security information to the MCU. The Flash security state is defined by the SEC bits of the FSEC register (see Table 28-12). During reset, the Flash module initializes the FSEC register using data read from the security byte of the Flash configuration field at global address 0x7F_FF0F.

The security state out of reset can be permanently changed by programming the security byte of the Flash configuration field. This assumes that you are starting from a mode where the necessary P-Flash erase and program commands are available and that the upper region of the P-Flash is unprotected. If the Flash security byte is successfully programmed, its new value will take affect after the next MCU reset.

The following subsections describe these security-related subjects:

- Unsecuring the MCU using Backdoor Key Access
- Unsecuring the MCU in Special Single Chip Mode using BDM
- Mode and Security Effects on Flash Command Availability

### 28.5.1 Unsecuring the MCU using Backdoor Key Access

The MCU may be unsecured by using the backdoor key access feature which requires knowledge of the contents of the backdoor keys (four 16-bit words programmed at addresses 0x7F_FF00–0x7F_FF07). If the KEYEN[1:0] bits are in the enabled state (see Section 28.3.2.2), the Verify Backdoor Access Key command (see Section 28.4.2.12) allows the user to present four prospective keys for comparison to the keys stored in the Flash memory via the Memory Controller. If the keys presented in the Verify Backdoor Access Key command match the backdoor keys stored in the Flash memory, the SEC bits in the FSEC register (see Table 28-12) will be changed to unsecure the MCU. Key values of 0x0000 and 0xFFFF are not permitted as backdoor keys. While the Verify Backdoor Access Key command is active, P-Flash block 0 will not be available for read access and will return invalid data.

The user code stored in the P-Flash memory must have a method of receiving the backdoor keys from an external stimulus. This external stimulus would typically be through one of the on-chip serial ports.

If the KEYEN[1:0] bits are in the enabled state (see Section 28.3.2.2), the MCU can be unsecured by the backdoor key access sequence described below:

1. Follow the command sequence for the Verify Backdoor Access Key command as explained in Section 28.4.2.12
2. If the Verify Backdoor Access Key command is successful, the MCU is unsecured and the SEC[1:0] bits in the FSEC register are forced to the unsecure state of 10

The Verify Backdoor Access Key command is monitored by the Memory Controller and an illegal key will prohibit future use of the Verify Backdoor Access Key command. A reset of the MCU is the only method to re-enable the Verify Backdoor Access Key command.

After the backdoor keys have been correctly matched, the MCU will be unsecured. After the MCU is unsecured, the sector containing the Flash security byte can be erased and the Flash security byte can be reprogrammed to the unsecure state, if desired.

In the unsecure state, the user has full control of the contents of the backdoor keys by programming addresses 0x7F_FF00–0x7F_FF07 in the Flash configuration field.

The security as defined in the Flash security byte (0x7F_FF0F) is not changed by using the Verify Backdoor Access Key command sequence. The backdoor keys stored in addresses 0x7F_FF00–0x7F_FF07 are unaffected by the Verify Backdoor Access Key command sequence. After the next reset of the MCU, the security state of the Flash module is determined by the Flash security byte

(0x7F_FF0F). The Verify Backdoor Access Key command sequence has no effect on the program and erase protections defined in the Flash protection register, FPROT.

### 28.5.2 Unsecuring the MCU in Special Single Chip Mode using BDM

The MCU can be unsecured in special single chip mode by erasing the P-Flash and D-Flash memory by one of the following methods:

- Reset the MCU into special single chip mode, delay while the erase test is performed by the BDM, send BDM commands to disable protection in the P-Flash and D-Flash memory, and execute the Erase All Blocks command write sequence to erase the P-Flash and D-Flash memory.
- Reset the MCU into special expanded wide mode, disable protection in the P-Flash and D-Flash memory and run code from external memory to execute the Erase All Blocks command write sequence to erase the P-Flash and D-Flash memory.

After the CCIF flag sets to indicate that the Erase All Blocks operation has completed, reset the MCU into special single chip mode. The BDM will execute the Erase Verify All Blocks command write sequence to verify that the P-Flash and D-Flash memory is erased. If the P-Flash and D-Flash memory are verified as erased the MCU will be unsecured. All BDM commands will be enabled and the Flash security byte may be programmed to the unsecure state by the following method:

- Send BDM commands to execute a ‘Program P-Flash’ command sequence to program the Flash security byte to the unsecured state and reset the MCU.

### 28.5.3 Mode and Security Effects on Flash Command Availability

The availability of Flash module commands depends on the MCU operating mode and security state as shown in Table 28-30.

## 28.6 Initialization

On each system reset the Flash module executes a reset sequence which establishes initial values for the Flash Block Configuration Parameters, the FPROT and DFPROT protection registers, and the FOPT and FSEC registers. The Flash module reverts to built-in default values that leave the module in a fully protected and secured state if errors are encountered during execution of the reset sequence. If a double bit fault is detected during the reset sequence, both MGSTAT bits in the FSTAT register will be set. The ACCERR bit in the FSTAT register is set if errors are encountered while initializing the EEE buffer ram during the reset sequence.

CCIF remains clear throughout the reset sequence. The Flash module holds off all CPU access for the initial portion of the reset sequence. While Flash reads are possible when the hold is removed, writes to the FCCOBIX, FCCOBHI, and FCCOBLO registers are ignored to prevent command activity while the Memory Controller remains busy. Completion of the reset sequence is marked by setting CCIF high which enables writes to the FCCOBIX, FCCOBHI, and FCCOBLO registers to launch any available Flash command.

If a reset occurs while any Flash command is in progress, that command will be immediately aborted. The state of the word being programmed or the sector/block being erased is not guaranteed.

# Chapter 29
# 1024 KByte Flash Module (S12XFTM1024K5V2)

**Table 29-1. Revision History**

| Revision Number | Revision Date | Sections Affected | Description of Changes |
|---|---|---|---|
| V02.08 | 14 Nov 2007 | 29.5.2/29-1200<br><br>29.4.2/29-1176<br><br><br>29.4.2.8/29-1182 | - Changed terminology from ‘word program’ to “Program P-Flash’ in the BDM unsecuring description, Section 29.5.2<br>- Added requirement that user not write any Flash module register during execution of commands ‘Erase All Blocks’, Section 29.4.2.8, and ‘Unsecure Flash’, Section 29.4.2.11<br>- Added statement that security is released upon successful completion of command ‘Erase All Blocks’, Section 29.4.2.8 |
| V02.09 | 19 Dec 2007 | 29.4.2/29-1176 | - Corrected Error Handling table for Full Partition D-Flash, Partition D-Flash, and EEPROM Emulation Query commands |
| V02.10 | 25 Sep 2009 | 29.1/29-1140<br>29.3.2.1/29-1152<br>29.4.2.4/29-1179<br><br>29.4.2.7/29-1181<br><br>29.4.2.12/29-1185<br><br>29.4.2.12/29-1185<br>29.4.2.12/29-1185<br>29.4.2.20/29-1194<br><br><br>29.3.2/29-1150<br>29.3.2.1/29-1152<br>29.4.1.2/29-1171<br>29.6/29-1200 | - Clarify single bit fault correction for P-Flash phrase<br>- Expand FDIV vs OSCCLK Frequency table<br>- Add statement concerning code runaway when executing Read Once command from Flash block containing associated fields<br>- Add statement concerning code runaway when executing Program Once command from Flash block containing associated fields<br>- Add statement concerning code runaway when executing Verify Backdoor Access Key command from Flash block containing associated fields<br>- Relate Key 0 to associated Backdoor Comparison Key address<br>- Change “power down reset” to “reset”<br>- Add ACCERR condition for Disable EEPROM Emulation command<br>The following changes were made to clarify module behavior related to Flash register access during reset sequence and while Flash commands are active:<br>- Add caution concerning register writes while command is active<br>- Writes to FCLKDIV are allowed during reset sequence while CCIF is clear<br>- Add caution concerning register writes while command is active<br>- Writes to FCCOBIX, FCCOBHI, FCCOBLO registers are ignored during reset sequence |

## 29.1 Introduction

The FTM1024K5 module implements the following:

- 1024 Kbytes of P-Flash (Program Flash) memory, consisting of 5 physical Flash blocks, intended primarily for nonvolatile code storage
- 32 Kbytes of D-Flash (Data Flash) memory, consisting of 1 physical Flash block, that can be used as nonvolatile storage to support the built-in hardware scheme for emulated EEPROM, as basic Flash memory primarily intended for nonvolatile data storage, or as a combination of both
- 4 Kbytes of buffer RAM, consisting of 1 physical RAM block, that can be used as emulated EEPROM using a built-in hardware scheme, as basic RAM, or as a combination of both

The Flash memory is ideal for single-supply applications allowing for field reprogramming without requiring external high voltage sources for program or erase operations. The Flash module includes a memory controller that executes commands to modify Flash memory contents or configure module resources for emulated EEPROM operation. The user interface to the memory controller consists of the indexed Flash Common Command Object (FCCOB) register which is written to with the command, global address, data, and any required command parameters. The memory controller must complete the execution of a command before the FCCOB register can be written to with a new command.

**CAUTION**

A Flash word or phrase must be in the erased state before being programmed. Cumulative programming of bits within a Flash word or phrase is not allowed.

The RAM and Flash memory may be read as bytes, aligned words, or misaligned words. Read access time is one bus cycle for bytes and aligned words, and two bus cycles for misaligned words. For Flash memory, an erased bit reads 1 and a programmed bit reads 0.

It is not possible to read from a Flash block while any command is executing on that specific Flash block. It is possible to read from a Flash block while a command is executing on a different Flash block.

Both P-Flash and D-Flash memories are implemented with Error Correction Codes (ECC) that can resolve single bit faults and detect double bit faults. For P-Flash memory, the ECC implementation requires that programming be done on an aligned 8 byte basis (a Flash phrase). Since P-Flash memory is always read by phrase, only one single bit fault in the phrase containing the byte or word accessed will be corrected.

### 29.1.1 Glossary

**Buffer RAM** — The buffer RAM constitutes the volatile memory store required for EEE. Memory space in the buffer RAM not required for EEE can be partitioned to provide volatile memory space for applications.

**Command Write Sequence** — An MCU instruction sequence to execute built-in algorithms (including program and erase) on the Flash memory.

**D-Flash Memory** — The D-Flash memory constitutes the nonvolatile memory store required for EEE. Memory space in the D-Flash memory not required for EEE can be partitioned to provide nonvolatile memory space for applications.

**D-Flash Sector** — The D-Flash sector is the smallest portion of the D-Flash memory that can be erased. The D-Flash sector consists of four 64 byte rows for a total of 256 bytes.

**EEE (Emulated EEPROM)** — A method to emulate the small sector size features and endurance characteristics associated with an EEPROM.

**EEE IFR** — Nonvolatile information register located in the D-Flash block that contains data required to partition the D-Flash memory and buffer RAM for EEE. The EEE IFR is visible in the global memory map by setting the EEEIFRON bit in the MMCCTL1 register.

**NVM Command Mode** — An NVM mode using the CPU to setup the FCCOB register to pass parameters required for Flash command execution.

**Phrase** — An aligned group of four 16-bit words within the P-Flash memory. Each phrase includes eight ECC bits for single bit fault correction and double bit fault detection within the phrase.

**P-Flash Memory** — The P-Flash memory constitutes the main nonvolatile memory store for applications.

**P-Flash Sector** — The P-Flash sector is the smallest portion of the P-Flash memory that can be erased. Each P-Flash sector contains 1024 bytes.

**Program IFR** — Nonvolatile information register located in the P-Flash block that contains the Device ID, Version ID, and the Program Once field. The Program IFR is visible in the global memory map by setting the PGMIFRON bit in the MMCCTL1 register.

## 29.1.2 Features

### 29.1.2.1 P-Flash Features

- 1024 Kbytes of P-Flash memory composed of three 256 Kbyte Flash blocks and two 128 Kbyte Flash blocks. The 256 Kbyte Flash block consists of two 128 Kbyte sections each divided into 128 sectors of 1024 bytes. The 128 Kbyte Flash blocks are each divided into 128 sectors of 1024 bytes.
- Single bit fault correction and double bit fault detection within a 64-bit phrase during read operations
- Automated program and erase algorithm with verify and generation of ECC parity bits
- Fast sector erase and phrase program operation
- Ability to program up to one phrase in each P-Flash block simultaneously
- Flexible protection scheme to prevent accidental program or erase of P-Flash memory

#### 29.1.2.2 D-Flash Features

- Up to 32 Kbytes of D-Flash memory with 256 byte sectors for user access
- Dedicated commands to control access to the D-Flash memory over EEE operation
- Single bit fault correction and double bit fault detection within a word during read operations
- Automated program and erase algorithm with verify and generation of ECC parity bits
- Fast sector erase and word program operation
- Ability to program up to four words in a burst sequence

#### 29.1.2.3 Emulated EEPROM Features

- Up to 4 Kbytes of emulated EEPROM (EEE) accessible as 4 Kbytes of RAM
- Flexible protection scheme to prevent accidental program or erase of data
- Automatic EEE file handling using an internal Memory Controller
- Automatic transfer of valid EEE data from D-Flash memory to buffer RAM on reset
- Ability to monitor the number of outstanding EEE related buffer RAM words left to be programmed into D-Flash memory
- Ability to disable EEE operation and allow priority access to the D-Flash memory
- Ability to cancel all pending EEE operations and allow priority access to the D-Flash memory

#### 29.1.2.4 User Buffer RAM Features

- Up to 4 Kbytes of RAM for user access

#### 29.1.2.5 Other Flash Module Features

- No external high-voltage power supply required for Flash memory program and erase operations
- Interrupt generation on Flash command completion and Flash error detection
- Security mechanism to prevent unauthorized access to the Flash memory

### 29.1.3 Block Diagram

The block diagram of the Flash module is shown in Figure 29-1.

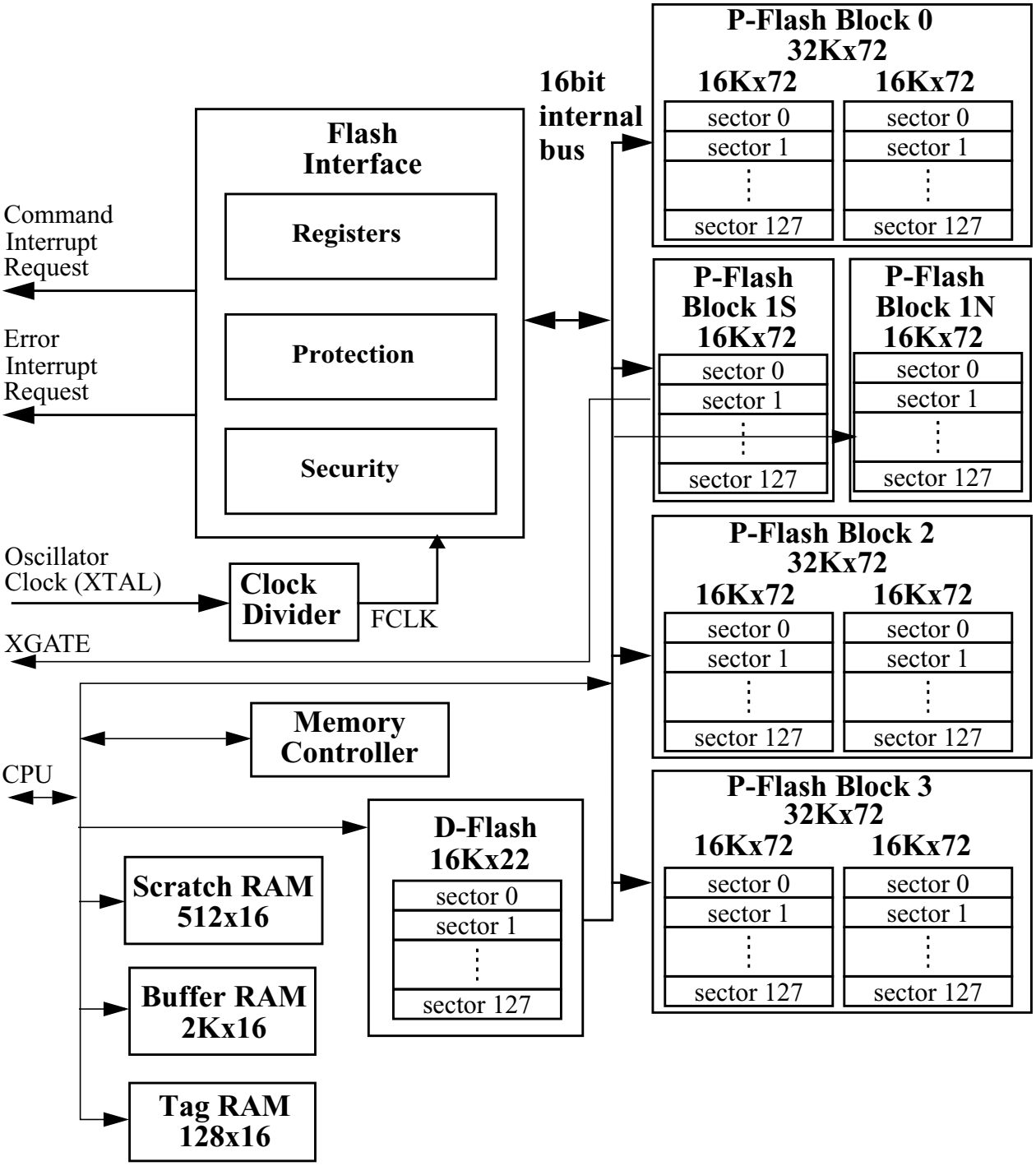



**Figure 29-1. FTM1024K5 Block Diagram**

## 29.2 External Signal Description

The Flash module contains no signals that connect off-chip.

## 29.3 Memory Map and Registers

This section describes the memory map and registers for the Flash module. Read data from unimplemented memory space in the Flash module is undefined. Write access to unimplemented or reserved memory space in the Flash module will be ignored by the Flash module.

### 29.3.1 Module Memory Map

The S12X architecture places the P-Flash memory between global addresses 0x70_0000 and 0x7F_FFFF as shown in Table 29-2. The P-Flash memory map is shown in Figure 29-2.

**Table 29-2. P-Flash Memory Addressing**

| Global Address | Size (Bytes) | Description |
|---|---|---|
| 0x7C_0000 – 0x7F_FFFF | 256 K | P-Flash Block 0<br>Contains Flash Configuration Field (see Table 29-3) |
| 0x7A_0000 – 0x7B_FFFF | 128 K | P-Flash Block 1N |
| 0x78_0000 – 0x79_FFFF | 128 K | P-Flash Block 1S |
| 0x74_0000 – 0x77_FFFF | 256 K | P-Flash Block 2 |
| 0x70_0000 – 0x73_FFFF | 256 K | P-Flash Block 3 |

The FPROT register, described in Section 29.3.2.9, can be set to protect regions in the Flash memory from accidental program or erase. Three separate memory regions, one growing upward from global address 0x7F_8000 in the Flash memory (called the lower region), one growing downward from global address 0x7F_FFFF in the Flash memory (called the higher region), and the remaining addresses in the Flash memory, can be activated for protection. The Flash memory addresses covered by these protectable regions are shown in the P-Flash memory map. The higher address region is mainly targeted to hold the boot loader code since it covers the vector space. Default protection settings as well as security information that allows the MCU to restrict access to the Flash module are stored in the Flash configuration field as described in Table 29-3.

**Table 29-3. Flash Configuration Field(1)**

| Global Address | Size (Bytes) | Description |
|---|---|---|
| 0x7F_FF00 – 0x7F_FF07 | 8 | Backdoor Comparison Key<br>Refer to Section 29.4.2.12, “Verify Backdoor Access Key Command,” and Section 29.5.1, “Unsecuring the MCU using Backdoor Key Access” |
| 0x7F_FF08 – 0x7F_FF0B(2) | 4 | Reserved |
| 0x7F_FF0C[2] | 1 | P-Flash Protection byte.<br>Refer to Section 29.3.2.9, “P-Flash Protection Register (FPROT)” |
| 0x7F_FF0D[2] | 1 | EEE Protection byte<br>Refer to Section 29.3.2.10, “EEE Protection Register (EPROT)” |

**Table 29-3. Flash Configuration Field[1]**

| Global Address | Size (Bytes) | Description |
|---|---|---|
| 0x7F_FF0E[2] | 1 | Flash Nonvolatile byte<br>Refer to Section 29.3.2.14, “Flash Option Register (FOPT)” |
| 0x7F_FF0F[2] | 1 | Flash Security byte<br>Refer to Section 29.3.2.2, “Flash Security Register (FSEC)” |

1. Older versions may have swapped protection byte addresses
2. 0x7FF08 - 0x7F_FF0F form a Flash phrase and must be programmed in a single command write sequence. Each byte in the 0x7F_FF08 - 0x7F_FF0B reserved field should be programmed to 0xFF.

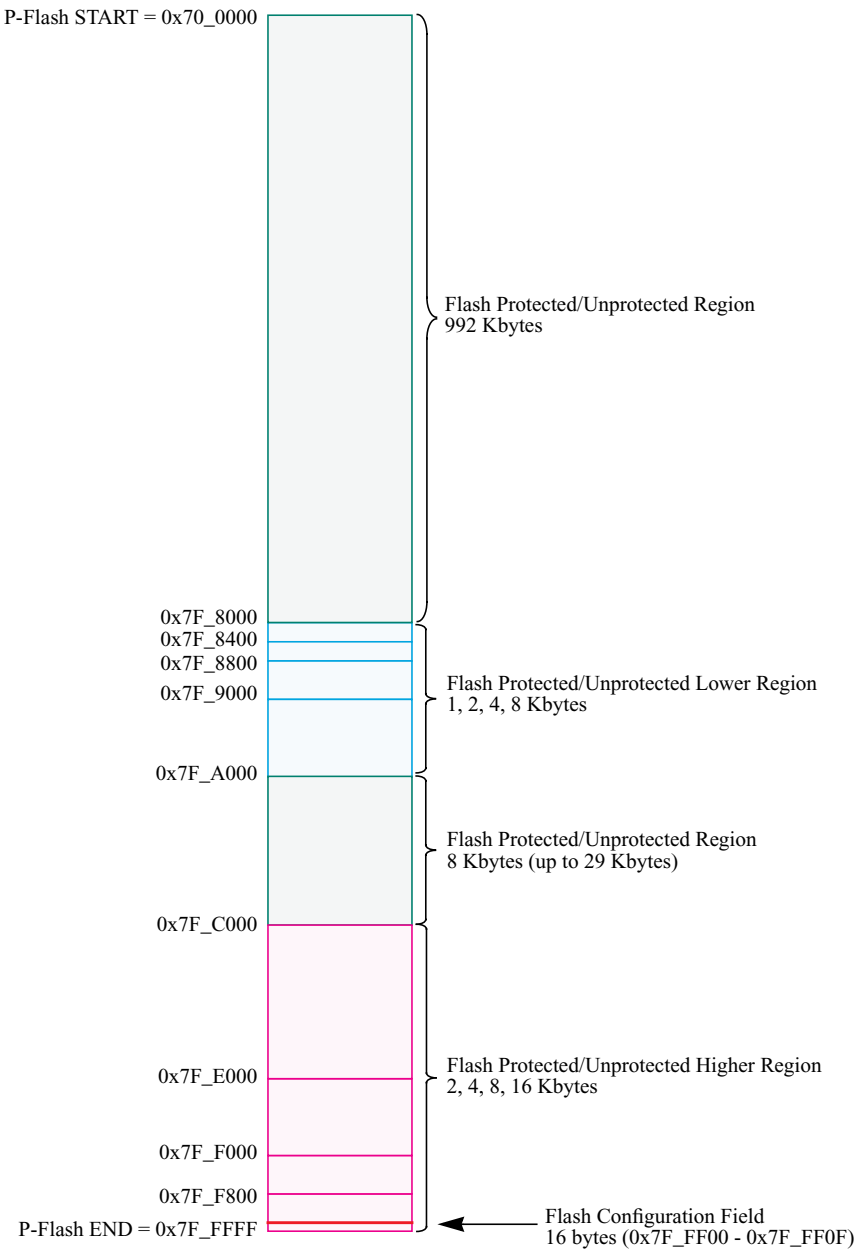



**Figure 29-2. P-Flash Memory Map**

**Table 29-4. Program IFR Fields**

| Global Address (PGMIFRON) | Size (Bytes) | Field Description |
|---|---|---|
| 0x40_0000 – 0x40_0007 | 8 | Device ID |
| 0x40_0008 – 0x40_00E7 | 224 | Reserved |
| 0x40_00E8 – 0x40_00E9 | 2 | Version ID |
| 0x40_00EA – 0x40_00FF | 22 | Reserved |
| 0x40_0100 – 0x40_013F | 64 | Program Once Field<br>Refer to Section 29.4.2.7, “Program Once Command” |
| 0x40_0140 – 0x40_01FF | 192 | Reserved |

**Table 29-5. P-Flash IFR Accessibility**

| Global Address (PGMIFRON) | Size (Bytes) | Accessed From |
|---|---|---|
| 0x40_0000 – 0x40_01FF | 512 | XBUS0 (PBLK0S)(1) |
| 0x40_0200 – 0x40_03FF | 512 | Unimplemented |
| 0x40_0400 – 0x40_05FF | 512 | XBUS0 (PBLK1N) |
| 0x40_0600 – 0x40_07FF | 512 | XBUS1 (PBLK1S) |
| 0x40_0800 – 0x40_09FF | 512 | XBUS0 (PBLK2S) |
| 0x40_0A00 – 0x40_0BFF | 512 | Unimplemented |
| 0x40_0C00 – 0x40_0DFF | 512 | XBUS0 (PBLK3S) |
| 0x40_0E00 – 0x40_0FFF | 512 | Unimplemented |

1. Refer to Table 29-4 for more details.

**Table 29-6. EEE Resource Fields**

| Global Address | Size (Bytes) | Description |
|---|---|---|
| 0x10_0000 – 0x10_7FFF | 32,768 | D-Flash Memory (User and EEE) |
| 0x10_8000 – 0x11_FFFF | 98,304 | Reserved |
| 0x12_0000 – 0x12_007F | 128 | EEE Nonvolatile Information Register (EEEIFRON(1) = 1) |
| 0x12_0080 – 0x12_0FFF | 3,968 | Reserved |
| 0x12_1000 – 0x12_1EFF | 3,840 | Reserved |
| 0x12_1F00 – 0x12_1FFF | 256 | EEE Tag RAM (TMGRAMON[1] = 1) |
| 0x12_2000 – 0x12_3BFF | 7,168 | Reserved |
| 0x12_3C00 – 0x12_3FFF | 1,024 | Memory Controller Scratch RAM (TMGRAMON[1] = 1) |
| 0x12_4000 – 0x12_DFFF | 40,960 | Reserved |
| 0x12_E000 – 0x12_FFFF | 8,192 | Reserved |
| 0x13_0000 – 0x13_EFFF | 61,440 | Reserved |
| 0x13_F000 – 0x13_FFFF | 4,096 | Buffer RAM (User and EEE) |

1. MMCCTL1 register bit

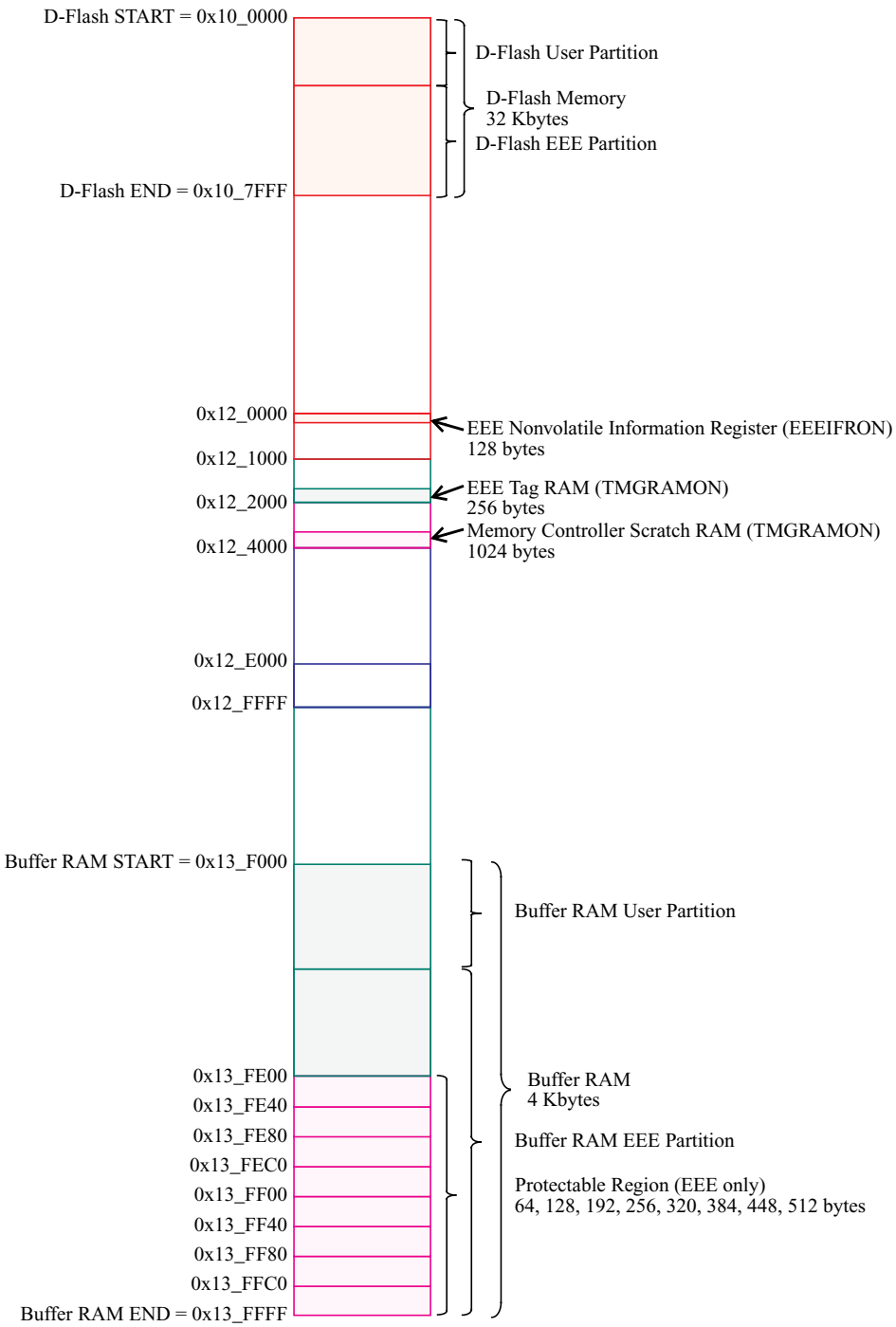



**Figure 29-3. EEE Resource Memory Map**

The Full Partition D-Flash command (see Section 29.4.2.15) is used to program the EEE nonvolatile information register fields where address 0x12_0000 defines the D-Flash partition for user access and address 0x12_0004 defines the buffer RAM partition for EEE operations.

**Table 29-7. EEE Nonvolatile Information Register Fields**

| Global Address (EEEIFRON) | Size (Bytes) | Description |
|---|---|---|
| 0x12_0000 – 0x12_0001 | 2 | D-Flash User Partition (DFPART)<br>Refer to Section 29.4.2.15, “Full Partition D-Flash Command” |
| 0x12_0002 – 0x12_0003 | 2 | D-Flash User Partition (duplicate[1]) |
| 0x12_0004 – 0x12_0005 | 2 | Buffer RAM EEE Partition (ERPART)<br>Refer to Section 29.4.2.15, “Full Partition D-Flash Command” |
| 0x12_0006 – 0x12_0007 | 2 | Buffer RAM EEE Partition (duplicate[1]) |
| 0x12_0008 – 0x12_007F | 120 | Reserved |

1. Duplicate value used if primary value generates a double bit fault when read during the reset sequence.

## 29.3.2 Register Descriptions

The Flash module contains a set of 20 control and status registers located between Flash module base + 0x0000 and 0x0013. A summary of the Flash module registers is given in Figure 29-4 with detailed descriptions in the following subsections.

**CAUTION**

Writes to any Flash register must be avoided while a Flash command is active (CCIF=0) to prevent corruption of Flash register contents and Memory Controller behavior.

| Address & Name | | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|
| 0x0000 FCLKDIV | R | FDIVLD | FDIV6 | FDIV5 | FDIV4 | FDIV3 | FDIV2 | FDIV1 | FDIV0 |
| | W | | | | | | | | |
| 0x0001 FSEC | R | KEYEN1 | KEYEN0 | RNV5 | RNV4 | RNV3 | RNV2 | SEC1 | SEC0 |
| | W | | | | | | | | |
| 0x0002 FCCOBIX | R | 0 | 0 | 0 | 0 | 0 | CCOBIX2 | CCOBIX1 | CCOBIX0 |
| | W | | | | | | | | |
| 0x0003 FECCRIX | R | 0 | 0 | 0 | 0 | 0 | ECCRIX2 | ECCRIX1 | ECCRIX0 |
| | W | | | | | | | | |
| 0x0004 FCNFG | R | CCIE | 0 | 0 | IGNSF | 0 | 0 | FDFD | FSFD |
| | W | | | | | | | | |
| 0x0005 FERCNFG | R | ERSERIE | PGMERIE | 0 | EPVIOLIE | ERSVIE1 | ERSVIE0 | DFDIE | SFDIE |
| | W | | | | | | | | |

**Figure 29-4. FTM1024K5 Register Summary**

| Address & Name | | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|
| 0x0006 FSTAT | R | CCIF | 0 | ACCERR | FPVIOL | MGBUSY | RSVD | MGSTAT1 | MGSTAT0 |
| | W | | | | | | | | |
| 0x0007 FERSTAT | R | ERSERIF | PGMERIF | 0 | EPVIOLIF | ERSVIF1 | ERSVIF0 | DFDIF | SFDIF |
| | W | | | | | | | | |
| 0x0008 FPROT | R | FPOPEN | RNV6 | FPHDIS | FPHS1 | FPHS0 | FPLDIS | FPLS1 | FPLS0 |
| | W | | | | | | | | |
| 0x0009 EPROT | R | EPOPEN | RNV6 | RNV5 | RNV4 | EPDIS | EPS2 | EPS1 | EPS0 |
| | W | | | | | | | | |
| 0x000A FCCOBHI | R | CCOB15 | CCOB14 | CCOB13 | CCOB12 | CCOB11 | CCOB10 | CCOB9 | CCOB8 |
| | W | | | | | | | | |
| 0x000B FCCOBLO | R | CCOB7 | CCOB6 | CCOB5 | CCOB4 | CCOB3 | CCOB2 | CCOB1 | CCOB0 |
| | W | | | | | | | | |
| 0x000C ETAGHI | R | ETAG15 | ETAG14 | ETAG13 | ETAG12 | ETAG11 | ETAG10 | ETAG9 | ETAG8 |
| | W | | | | | | | | |
| 0x000D ETAGLO | R | ETAG7 | ETAG6 | ETAG5 | ETAG4 | ETAG3 | ETAG2 | ETAG1 | ETAG0 |
| | W | | | | | | | | |
| 0x000E FECCRHI | R | ECCR15 | ECCR14 | ECCR13 | ECCR12 | ECCR11 | ECCR10 | ECCR9 | ECCR8 |
| | W | | | | | | | | |
| 0x000F FECCRLO | R | ECCR7 | ECCR6 | ECCR5 | ECCR4 | ECCR3 | ECCR2 | ECCR1 | ECCR0 |
| | W | | | | | | | | |
| 0x0010 FOPT | R | NV7 | NV6 | NV5 | NV4 | NV3 | NV2 | NV1 | NV0 |
| | W | | | | | | | | |
| 0x0011 FRSV0 | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | W | | | | | | | | |
| 0x0012 FRSV1 | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | W | | | | | | | | |
| 0x0013 FRSV2 | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | W | | | | | | | | |

**Figure 29-4. FTM1024K5 Register Summary (continued)**

| Address & Name | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | = Unimplemented or Reserved | | | | | | | |

**Figure 29-4. FTM1024K5 Register Summary (continued)**

### 29.3.2.1 Flash Clock Divider Register (FCLKDIV)

The FCLKDIV register is used to control timed events in program and erase algorithms.

Offset Module Base + 0x0000

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | FDIVLD | FDIV[6:0] | | | | | | |
| W | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented or Reserved

**Figure 29-5. Flash Clock Divider Register (FCLKDIV)**

All bits in the FCLKDIV register are readable, bits 6–0 are write once and bit 7 is not writable.

**Table 29-8. FCLKDIV Field Descriptions**

| Field | Description |
|---|---|
| 7<br>FDIVLD | **Clock Divider Loaded**<br>0 FCLKDIV register has not been written<br>1 FCLKDIV register has been written since the last reset |
| 6–0<br>FDIV[6:0] | **Clock Divider Bits** — FDIV[6:0] must be set to effectively divide OSCCLK down to generate an internal Flash clock, FCLK, with a target frequency of 1 MHz for use by the Flash module to control timed events during program and erase algorithms. Table 29-9 shows recommended values for FDIV[6:0] based on OSCCLK frequency. Please refer to Section 29.4.1, “Flash Command Operations,” for more information. |

**CAUTION**

The FCLKDIV register should never be written while a Flash command is executing (CCIF=0). The FCLKDIV register is writable during the Flash reset sequence even though CCIF is clear.

**Table 29-9. FDIV vs OSCCLK Frequency**

| OSCCLK Frequency (MHz) | | FDIV[6:0] | OSCCLK Frequency (MHz) | | FDIV[6:0] | OSCCLK Frequency (MHz) | | FDIV[6:0] |
|---|---|---|---|---|---|---|---|---|
| MIN(1) | MAX(2) | | MIN[1] | MAX[2] | | MIN[1] | MAX[2] | |
| | | | 33.60 | 34.65 | 0x20 | 67.20 | 68.25 | 0x40 |
| 1.60 | 2.10 | 0x01 | 34.65 | 35.70 | 0x21 | 68.25 | 69.30 | 0x41 |
| 2.40 | 3.15 | 0x02 | 35.70 | 36.75 | 0x22 | 69.30 | 70.35 | 0x42 |
| 3.20 | 4.20 | 0x03 | 36.75 | 37.80 | 0x23 | 70.35 | 71.40 | 0x43 |
| 4.20 | 5.25 | 0x04 | 37.80 | 38.85 | 0x24 | 71.40 | 72.45 | 0x44 |
| 5.25 | 6.30 | 0x05 | 38.85 | 39.90 | 0x25 | 72.45 | 73.50 | 0x45 |
| 6.30 | 7.35 | 0x06 | 39.90 | 40.95 | 0x26 | 73.50 | 74.55 | 0x46 |
| 7.35 | 8.40 | 0x07 | 40.95 | 42.00 | 0x27 | 74.55 | 75.60 | 0x47 |
| 8.40 | 9.45 | 0x08 | 42.00 | 43.05 | 0x28 | 75.60 | 76.65 | 0x48 |
| 9.45 | 10.50 | 0x09 | 43.05 | 44.10 | 0x29 | 76.65 | 77.70 | 0x49 |
| 10.50 | 11.55 | 0x0A | 44.10 | 45.15 | 0x2A | 77.70 | 78.75 | 0x4A |
| 11.55 | 12.60 | 0x0B | 45.15 | 46.20 | 0x2B | 78.75 | 79.80 | 0x4B |
| 12.60 | 13.65 | 0x0C | 46.20 | 47.25 | 0x2C | 79.80 | 80.85 | 0x4C |
| 13.65 | 14.70 | 0x0D | 47.25 | 48.30 | 0x2D | 80.85 | 81.90 | 0x4D |
| 14.70 | 15.75 | 0x0E | 48.30 | 49.35 | 0x2E | 81.90 | 82.95 | 0x4E |
| 15.75 | 16.80 | 0x0F | 49.35 | 50.40 | 0x2F | 82.95 | 84.00 | 0x4F |
| 16.80 | 17.85 | 0x10 | 50.40 | 51.45 | 0x30 | 84.00 | 85.05 | 0x50 |
| 17.85 | 18.90 | 0x11 | 51.45 | 52.50 | 0x31 | 85.05 | 86.10 | 0x51 |
| 18.90 | 19.95 | 0x12 | 52.50 | 53.55 | 0x32 | 86.10 | 87.15 | 0x52 |
| 19.95 | 21.00 | 0x13 | 53.55 | 54.60 | 0x33 | 87.15 | 88.20 | 0x53 |
| 21.00 | 22.05 | 0x14 | 54.60 | 55.65 | 0x34 | 88.20 | 89.25 | 0x54 |
| 22.05 | 23.10 | 0x15 | 55.65 | 56.70 | 0x35 | 89.25 | 90.30 | 0x55 |
| 23.10 | 24.15 | 0x16 | 56.70 | 57.75 | 0x36 | 90.30 | 91.35 | 0x56 |
| 24.15 | 25.20 | 0x17 | 57.75 | 58.80 | 0x37 | 91.35 | 92.40 | 0x57 |
| 25.20 | 26.25 | 0x18 | 58.80 | 59.85 | 0x38 | 92.40 | 93.45 | 0x58 |
| 26.25 | 27.30 | 0x19 | 59.85 | 60.90 | 0x39 | 93.45 | 94.50 | 0x59 |
| 27.30 | 28.35 | 0x1A | 60.90 | 61.95 | 0x3A | 94.50 | 95.55 | 0x5A |
| 28.35 | 29.40 | 0x1B | 61.95 | 63.00 | 0x3B | 95.55 | 96.60 | 0x5B |
| 29.40 | 30.45 | 0x1C | 63.00 | 64.05 | 0x3C | 96.60 | 97.65 | 0x5C |
| 30.45 | 31.50 | 0x1D | 64.05 | 65.10 | 0x3D | 97.65 | 98.70 | 0x5D |
| 31.50 | 32.55 | 0x1E | 65.10 | 66.15 | 0x3E | 98.70 | 99.75 | 0x5E |
| 32.55 | 33.60 | 0x1F | 66.15 | 67.20 | 0x3F | 99.75 | 100.80 | 0x5F |

1. FDIV shown generates an FCLK frequency of >0.8 MHz

2. FDIV shown generates an FCLK frequency of 1.05 MHz

### 29.3.2.2 Flash Security Register (FSEC)

The FSEC register holds all bits associated with the security of the MCU and Flash module.

Offset Module Base + 0x0001

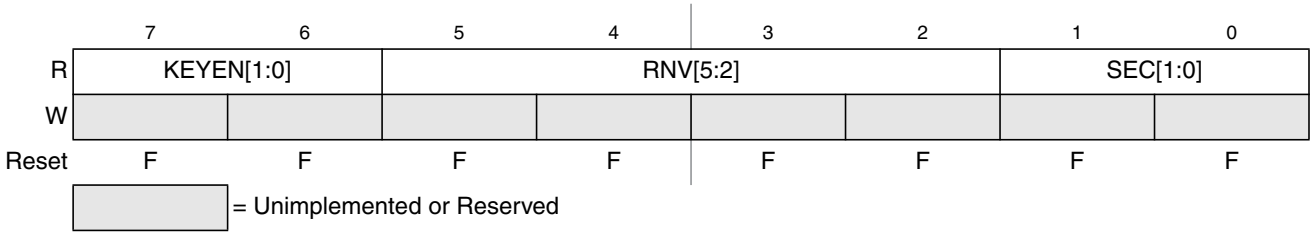



**Figure 29-6. Flash Security Register (FSEC)**

All bits in the FSEC register are readable but not writable.

During the reset sequence, the FSEC register is loaded with the contents of the Flash security byte in the Flash configuration field at global address 0x7F_FF0F located in P-Flash memory (see Table 29-3) as indicated by reset condition F in Figure 29-6. If a double bit fault is detected while reading the P-Flash phrase containing the Flash security byte during the reset sequence, all bits in the FSEC register will be set to leave the Flash module in a secured state with backdoor key access disabled.

**Table 29-10. FSEC Field Descriptions**

| Field | Description |
|---|---|
| 7–6<br>KEYEN[1:0] | **Backdoor Key Security Enable Bits** — The KEYEN[1:0] bits define the enabling of backdoor key access to the Flash module as shown in Table 29-11. |
| 5–2<br>RNV[5:2} | **Reserved Nonvolatile Bits** — The RNV bits should remain in the erased state for future enhancements. |
| 1–0<br>SEC[1:0] | **Flash Security Bits** — The SEC[1:0] bits define the security state of the MCU as shown in Table 29-12. If the Flash module is unsecured using backdoor key access, the SEC bits are forced to 10. |

**Table 29-11. Flash KEYEN States**

| KEYEN[1:0] | Status of Backdoor Key Access |
|---|---|
| 00 | DISABLED |
| 01 | DISABLED(1) |
| 10 | ENABLED |
| 11 | DISABLED |

1. Preferred KEYEN state to disable backdoor key access.

**Table 29-12. Flash Security States**

| SEC[1:0] | Status of Security |
|---|---|
| 00 | SECURED |
| 01 | SECURED(1) |
| 10 | UNSECURED |
| 11 | SECURED |

1. Preferred SEC state to set MCU to secured state.

The security function in the Flash module is described in Section 29.5.

### 29.3.2.3 Flash CCOB Index Register (FCCOBIX)

The FCCOBIX register is used to index the FCCOB register for Flash memory operations.

Offset Module Base + 0x0002

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | 0 | 0 | 0 | CCOBIX[2:0] | | |
| W | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented or Reserved

**Figure 29-7. FCCOB Index Register (FCCOBIX)**

CCOBIX bits are readable and writable while remaining bits read 0 and are not writable.

**Table 29-13. FCCOBIX Field Descriptions**

| Field | Description |
|---|---|
| 2–0 CCOBIX[1:0] | **Common Command Register Index**— The CCOBIX bits are used to select which word of the FCCOB register array is being read or written to. See Section 29.3.2.11, “Flash Common Command Object Register (FCCOB),” for more details. |

### 29.3.2.4 Flash ECCR Index Register (FECCRIX)

The FECCRIX register is used to index the FECCR register for ECC fault reporting.

Offset Module Base + 0x0003

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | 0 | 0 | 0 | ECCRIX[2:0] | | |
| W | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented or Reserved

**Figure 29-8. FECCR Index Register (FECCRIX)**

ECCRIX bits are readable and writable while remaining bits read 0 and are not writable.

**Table 29-14. FECCRIX Field Descriptions**

| Field | Description |
|---|---|
| 2-0<br>ECCRIX[2:0] | **ECC Error Register Index**— The ECCRIX bits are used to select which word of the FECCR register array is being read. See Section 29.3.2.13, “Flash ECC Error Results Register (FECCR),” for more details. |

### 29.3.2.5 Flash Configuration Register (FCNFG)

The FCNFG register enables the Flash command complete interrupt and forces ECC faults on Flash array read access from the CPU or XGATE.

Offset Module Base + 0x0004

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | CCIE | 0 | 0 | IGNSF | 0 | 0 | FDFD | FSFD |
| W | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented or Reserved

**Figure 29-9. Flash Configuration Register (FCNFG)**

CCIE, IGNSF, FDFD, and FSFD bits are readable and writable while remaining bits read 0 and are not writable.

**Table 29-15. FCNFG Field Descriptions**

| Field | Description |
|---|---|
| 7<br>CCIE | **Command Complete Interrupt Enable** — The CCIE bit controls interrupt generation when a Flash command has completed.<br>0 Command complete interrupt disabled<br>1 An interrupt will be requested whenever the CCIF flag in the FSTAT register is set (see Section 29.3.2.7) |
| 4<br>IGNSF | **Ignore Single Bit Fault** — The IGNSF controls single bit fault reporting in the FERSTAT register (see Section 29.3.2.8).<br>0 All single bit faults detected during array reads are reported<br>1 Single bit faults detected during array reads are not reported and the single bit fault interrupt will not be generated |

**Table 29-15. FCNFG Field Descriptions (continued)**

| Field | Description |
|---|---|
| 1<br>FDFD | **Force Double Bit Fault Detect** — The FDFD bit allows the user to simulate a double bit fault during Flash array read operations and check the associated interrupt routine. The FDFD bit is cleared by writing a 0 to FDFD. The FECCR registers will not be updated during the Flash array read operation with FDFD set unless an actual double bit fault is detected.<br>0 Flash array read operations will set the DFDIF flag in the FERSTAT register only if a double bit fault is detected<br>1 Any Flash array read operation will force the DFDIF flag in the FERSTAT register to be set (see Section 29.3.2.7) and an interrupt will be generated as long as the DFDIE interrupt enable in the FERCNFG register is set (see Section 29.3.2.6) |
| 0<br>FSFD | **Force Single Bit Fault Detect** — The FSFD bit allows the user to simulate a single bit fault during Flash array read operations and check the associated interrupt routine. The FSFD bit is cleared by writing a 0 to FSFD. The FECCR registers will not be updated during the Flash array read operation with FSFD set unless an actual single bit fault is detected.<br>0 Flash array read operations will set the SFDIF flag in the FERSTAT register only if a single bit fault is detected<br>1 Flash array read operation will force the SFDIF flag in the FERSTAT register to be set (see Section 29.3.2.7) and an interrupt will be generated as long as the SFDIE interrupt enable in the FERCNFG register is set (see Section 29.3.2.6) |

## 29.3.2.6 Flash Error Configuration Register (FERCNFG)

The FERCNFG register enables the Flash error interrupts for the FERSTAT flags.

Offset Module Base + 0x0005

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | ERSERIE | PGMERIE | 0 | EPVIOLIE | ERSVIE1 | ERSVIE0 | DFDIE | SFDIE |
| W | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented or Reserved

**Figure 29-10. Flash Error Configuration Register (FERCNFG)**

All assigned bits in the FERCNFG register are readable and writable.

**Table 29-16. FERCNFG Field Descriptions**

| Field | Description |
|---|---|
| 7<br>ERSERIE | **EEE Erase Error Interrupt Enable** — The ERSERIE bit controls interrupt generation when a failure is detected during an EEE erase operation.<br>0 ERSERIF interrupt disabled<br>1 An interrupt will be requested whenever the ERSERIF flag is set (see Section 29.3.2.8) |
| 6<br>PGMERIE | **EEE Program Error Interrupt Enable** — The PGMERIE bit controls interrupt generation when a failure is detected during an EEE program operation.<br>0 PGMERIF interrupt disabled<br>1 An interrupt will be requested whenever the PGMERIF flag is set (see Section 29.3.2.8) |
| 4<br>EPVIOLIE | **EEE Protection Violation Interrupt Enable** — The EPVIOLIE bit controls interrupt generation when a protection violation is detected during a write to the buffer RAM EEE partition.<br>0 EPVIOLIF interrupt disabled<br>1 An interrupt will be requested whenever the EPVIOLIF flag is set (see Section 29.3.2.8) |

**Table 29-16. FERCNFG Field Descriptions (continued)**

| Field | Description |
|---|---|
| 3<br>ERSVIE1 | **EEE Error Type 1 Interrupt Enable** — The ERSVIE1 bit controls interrupt generation when a change state error is detected during an EEE operation.<br>0 ERSVIF1 interrupt disabled<br>1 An interrupt will be requested whenever the ERSVIF1 flag is set (see Section 29.3.2.8) |
| 2<br>ERSVIE0 | **EEE Error Type 0 Interrupt Enable** — The ERSVIE0 bit controls interrupt generation when a sector format error is detected during an EEE operation.<br>0 ERSVIF0 interrupt disabled<br>1 An interrupt will be requested whenever the ERSVIF0 flag is set (see Section 29.3.2.8) |
| 1<br>DFDIE | **Double Bit Fault Detect Interrupt Enable** — The DFDIE bit controls interrupt generation when a double bit fault is detected during a Flash block read operation.<br>0 DFDIF interrupt disabled<br>1 An interrupt will be requested whenever the DFDIF flag is set (see Section 29.3.2.8) |
| 0<br>SFDIE | **Single Bit Fault Detect Interrupt Enable** — The SFDIE bit controls interrupt generation when a single bit fault is detected during a Flash block read operation.<br>0 SFDIF interrupt disabled whenever the SFDIF flag is set (see Section 29.3.2.8)<br>1 An interrupt will be requested whenever the SFDIF flag is set (see Section 29.3.2.8) |

### 29.3.2.7 Flash Status Register (FSTAT)

The FSTAT register reports the operational status of the Flash module.

Offset Module Base + 0x0006

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | CCIF | 0 | ACCERR | FPVIOL | MGBUSY | RSVD | MGSTAT[1:0] | |
| W | | | | | | | | |
| Reset | 1 | 0 | 0 | 0 | 0 | 0 | 0(1) | 0[1] |

= Unimplemented or Reserved

**Figure 29-11. Flash Status Register (FSTAT)**

1. Reset value can deviate from the value shown if a double bit fault is detected during the reset sequence (see Section 29.6).

CCIF, ACCERR, and FPVIOL bits are readable and writable, MGBUSY and MGSTAT bits are readable but not writable, while remaining bits read 0 and are not writable.

**Table 29-17. FSTAT Field Descriptions**

| Field | Description |
|---|---|
| 7<br>CCIF | **Command Complete Interrupt Flag** — The CCIF flag indicates that a Flash command has completed. The CCIF flag is cleared by writing a 1 to CCIF to launch a command and CCIF will stay low until command completion or command violation.<br>0 Flash command in progress<br>1 Flash command has completed |
| 5<br>ACCERR | **Flash Access Error Flag** — The ACCERR bit indicates an illegal access has occurred to the Flash memory caused by either a violation of the command write sequence (see Section 29.4.1.2) or issuing an illegal Flash command or when errors are encountered while initializing the EEE buffer ram during the reset sequence. While ACCERR is set, the CCIF flag cannot be cleared to launch a command. The ACCERR bit is cleared by writing a 1 to ACCERR. Writing a 0 to the ACCERR bit has no effect on ACCERR.<br>0 No access error detected<br>1 Access error detected |
| 4<br>FPVIOL | **Flash Protection Violation Flag** —The FPVIOL bit indicates an attempt was made to program or erase an address in a protected area of P-Flash memory during a command write sequence. The FPVIOL bit is cleared by writing a 1 to FPVIOL. Writing a 0 to the FPVIOL bit has no effect on FPVIOL. While FPVIOL is set, it is not possible to launch a command or start a command write sequence.<br>0 No protection violation detected<br>1 Protection violation detected |
| 3<br>MGBUSY | **Memory Controller Busy Flag** — The MGBUSY flag reflects the active state of the Memory Controller.<br>0 Memory Controller is idle<br>1 Memory Controller is busy executing a Flash command (CCIF = 0) or is handling internal EEE operations |
| 2<br>RSVD | **Reserved Bit** — This bit is reserved and always reads 0. |
| 1–0<br>MGSTAT[1:0] | **Memory Controller Command Completion Status Flag** — One or more MGSTAT flag bits are set if an error is detected during execution of a Flash command or during the Flash reset sequence. See Section 29.4.2, “Flash Command Description,” and Section 29.6, “Initialization” for details. |

### 29.3.2.8 Flash Error Status Register (FERSTAT)

The FERSTAT register reflects the error status of internal Flash operations.

Offset Module Base + 0x0007

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | ERSERIF | PGMERIF | 0 | EPVIOLIF | ERSVIF1 | ERSVIF0 | DFDIF | SFDIF |
| W | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented or Reserved

**Figure 29-12. Flash Error Status Register (FERSTAT)**

All flags in the FERSTAT register are readable and only writable to clear the flag.

**Table 29-18. FERSTAT Field Descriptions**

| Field | Description |
|---|---|
| 7<br>ERSERIF | **EEE Erase Error Interrupt Flag** — The setting of the ERSERIF flag occurs due to an error in a Flash erase command that resulted in the erase operation not being successful during EEE operations. The ERSERIF flag is cleared by writing a 1 to ERSERIF. Writing a 0 to the ERSERIF flag has no effect on ERSERIF. While ERSERIF is set, it is possible to write to the buffer RAM EEE partition but the data written will not be transferred to the D-Flash EEE partition.<br>0 Erase command successfully completed on the D-Flash EEE partition<br>1 Erase command failed on the D-Flash EEE partition |
| 6<br>PGMERIF | **EEE Program Error Interrupt Flag** — The setting of the PGMERIF flag occurs due to an error in a Flash program command that resulted in the program operation not being successful during EEE operations. The PGMERIF flag is cleared by writing a 1 to PGMERIF. Writing a 0 to the PGMERIF flag has no effect on PGMERIF. While PGMERIF is set, it is possible to write to the buffer RAM EEE partition but the data written will not be transferred to the D-Flash EEE partition.<br>0 Program command successfully completed on the D-Flash EEE partition<br>1 Program command failed on the D-Flash EEE partition |
| 4<br>EPVIOLIF | **EEE Protection Violation Interrupt Flag** —The setting of the EPVIOLIF flag indicates an attempt was made to write to a protected area of the buffer RAM EEE partition. The EPVIOLIF flag is cleared by writing a 1 to EPVIOLIF. Writing a 0 to the EPVIOLIF flag has no effect on EPVIOLIF. While EPVIOLIF is set, it is possible to write to the buffer RAM EEE partition as long as the address written to is not in a protected area.<br>0 No EEE protection violation<br>1 EEE protection violation detected |
| 3<br>ERSVIF1 | **EEE Error Interrupt 1 Flag** —The setting of the ERSVIF1 flag indicates that the memory controller was unable to change the state of a D-Flash EEE sector. The ERSVIF1 flag is cleared by writing a 1 to ERSVIF1. Writing a 0 to the ERSVIF1 flag has no effect on ERSVIF1. While ERSVIF1 is set, it is possible to write to the buffer RAM EEE partition but the data written will not be transferred to the D-Flash EEE partition.<br>0 No EEE sector state change error detected<br>1 EEE sector state change error detected |
| 2<br>ERSVIF0 | **EEE Error Interrupt 0 Flag** —The setting of the ERSVIF0 flag indicates that the memory controller was unable to format a D-Flash EEE sector for EEE use. The ERSVIF0 flag is cleared by writing a 1 to ERSVIF0. Writing a 0 to the ERSVIF0 flag has no effect on ERSVIF0. While ERSVIF0 is set, it is possible to write to the buffer RAM EEE partition but the data written will not be transferred to the D-Flash EEE partition.<br>0 No EEE sector format error detected<br>1 EEE sector format error detected |
| 1<br>DFDIF | **Double Bit Fault Detect Interrupt Flag** — The setting of the DFDIF flag indicates that a double bit fault was detected in the stored parity and data bits during a Flash array read operation or that a Flash array read operation was attempted on a Flash block that was under a Flash command operation. The DFDIF flag is cleared by writing a 1 to DFDIF. Writing a 0 to DFDIF has no effect on DFDIF.<br>0 No double bit fault detected<br>1 Double bit fault detected or an invalid Flash array read operation attempted |
| 0<br>SFDIF | **Single Bit Fault Detect Interrupt Flag** — With the IGNSF bit in the FCNFG register clear, the SFDIF flag indicates that a single bit fault was detected in the stored parity and data bits during a Flash array read operation or that a Flash array read operation was attempted on a Flash block that was under a Flash command operation. The SFDIF flag is cleared by writing a 1 to SFDIF. Writing a 0 to SFDIF has no effect on SFDIF.<br>0 No single bit fault detected<br>1 Single bit fault detected and corrected or an invalid Flash array read operation attempted |

### 29.3.2.9 P-Flash Protection Register (FPROT)

The FPROT register defines which P-Flash sectors are protected against program and erase operations.

Offset Module Base + 0x0008

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | FPOPEN | RNV6 | FPHDIS | FPHS[1:0] | | FPLDIS | FPLS[1:0] | |
| W | | | | | | | | |
| Reset | F | F | F | F | F | F | F | F |

= Unimplemented or Reserved

**Figure 29-13. Flash Protection Register (FPROT)**

The (unreserved) bits of the FPROT register are writable with the restriction that the size of the protected region can only be increased (see Section 29.3.2.9.1, “P-Flash Protection Restrictions,” and Table 29-23).

During the reset sequence, the FPROT register is loaded with the contents of the P-Flash protection byte in the Flash configuration field at global address 0x7F_FF0C located in P-Flash memory (see Table 29-3) as indicated by reset condition ‘F’ in Figure 29-13. To change the P-Flash protection that will be loaded during the reset sequence, the upper sector of the P-Flash memory must be unprotected, then the P-Flash protection byte must be reprogrammed. If a double bit fault is detected while reading the P-Flash phrase containing the P-Flash protection byte during the reset sequence, the FPOPEN bit will be cleared and remaining bits in the FPROT register will be set to leave the P-Flash memory fully protected.

Trying to alter data in any protected area in the P-Flash memory will result in a protection violation error and the FPVIOL bit will be set in the FSTAT register. The block erase of a P-Flash block is not possible if any of the P-Flash sectors contained in the same P-Flash block are protected.

**Table 29-19. FPROT Field Descriptions**

| Field | Description |
|---|---|
| 7<br>FPOPEN | **Flash Protection Operation Enable** — The FPOPEN bit determines the protection function for program or erase operations as shown in Table 29-20 for the P-Flash block.<br>0 When FPOPEN is clear, the FPHDIS and FPLDIS bits define unprotected address ranges as specified by the corresponding FPHS and FPLS bits<br>1 When FPOPEN is set, the FPHDIS and FPLDIS bits enable protection for the address range specified by the corresponding FPHS and FPLS bits |
| 6<br>RNV[6] | **Reserved Nonvolatile Bit** — The RNV bit should remain in the erased state for future enhancements. |
| 5<br>FPHDIS | **Flash Protection Higher Address Range Disable** — The FPHDIS bit determines whether there is a protected/unprotected area in a specific region of the P-Flash memory ending with global address 0x7F_FFFF.<br>0 Protection/Unprotection enabled<br>1 Protection/Unprotection disabled |
| 4–3<br>FPHS[1:0] | **Flash Protection Higher Address Size** — The FPHS bits determine the size of the protected/unprotected area in P-Flash memory as shown inTable 29-21. The FPHS bits can only be written to while the FPHDIS bit is set. |
| 2<br>FPLDIS | **Flash Protection Lower Address Range Disable** — The FPLDIS bit determines whether there is a protected/unprotected area in a specific region of the P-Flash memory beginning with global address 0x7F_8000.<br>0 Protection/Unprotection enabled<br>1 Protection/Unprotection disabled |
| 1–0<br>FPLS[1:0] | **Flash Protection Lower Address Size** — The FPLS bits determine the size of the protected/unprotected area in P-Flash memory as shown in Table 29-22. The FPLS bits can only be written to while the FPLDIS bit is set. |

**Table 29-20. P-Flash Protection Function**

| FPOPEN | FPHDIS | FPLDIS | Function[1] |
|---|---|---|---|
| 1 | 1 | 1 | No P-Flash Protection |
| 1 | 1 | 0 | Protected Low Range |
| 1 | 0 | 1 | Protected High Range |
| 1 | 0 | 0 | Protected High and Low Ranges |
| 0 | 1 | 1 | Full P-Flash Memory Protected |
| 0 | 1 | 0 | Unprotected Low Range |
| 0 | 0 | 1 | Unprotected High Range |
| 0 | 0 | 0 | Unprotected High and Low Ranges |

1. For range sizes, refer to Table 29-21 and Table 29-22.

**Table 29-21. P-Flash Protection Higher Address Range**

| FPHS[1:0] | Global Address Range | Protected Size |
|---|---|---|
| 00 | 0x7F_F800–0x7F_FFFF | 2 Kbytes |
| 01 | 0x7F_F000–0x7F_FFFF | 4 Kbytes |
| 10 | 0x7F_E000–0x7F_FFFF | 8 Kbytes |
| 11 | 0x7F_C000–0x7F_FFFF | 16 Kbytes |

**Table 29-22. P-Flash Protection Lower Address Range**

| FPLS[1:0] | Global Address Range | Protected Size |
|---|---|---|
| 00 | 0x7F_8000–0x7F_83FF | 1 Kbyte |
| 01 | 0x7F_8000–0x7F_87FF | 2 Kbytes |
| 10 | 0x7F_8000–0x7F_8FFF | 4 Kbytes |
| 11 | 0x7F_8000–0x7F_9FFF | 8 Kbytes |

All possible P-Flash protection scenarios are shown in Figure 29-14. Although the protection scheme is loaded from the Flash memory at global address 0x7F_FF0C during the reset sequence, it can be changed by the user. The P-Flash protection scheme can be used by applications requiring reprogramming in single chip mode while providing as much protection as possible if reprogramming is not required.

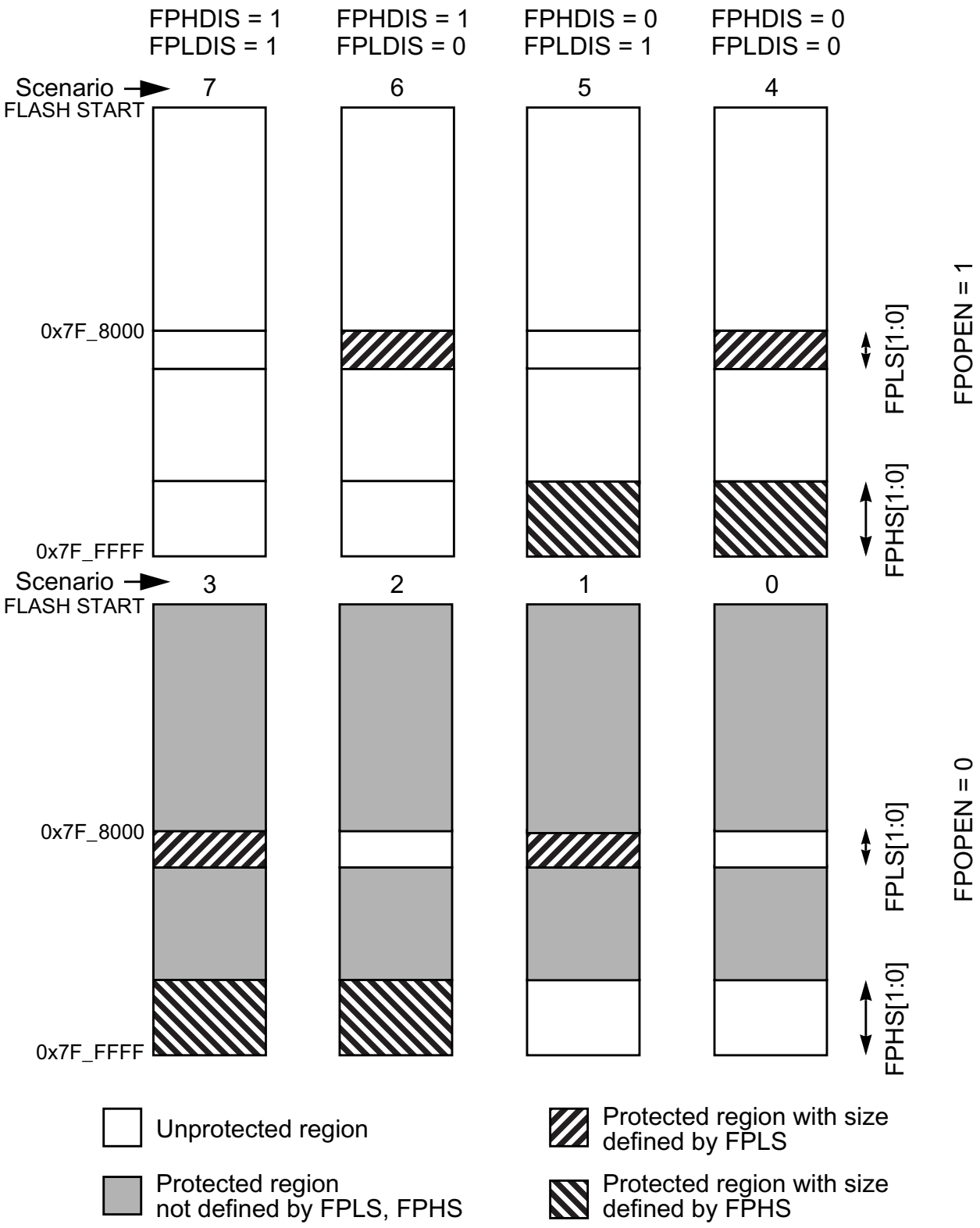



**Figure 29-14. P-Flash Protection Scenarios**

#### 29.3.2.9.1 P-Flash Protection Restrictions

The general guideline is that P-Flash protection can only be added and not removed. Table 29-23 specifies all valid transitions between P-Flash protection scenarios. Any attempt to write an invalid scenario to the FPROT register will be ignored. The contents of the FPROT register reflect the active protection scenario. See the FPHS and FPLS bit descriptions for additional restrictions.

**Table 29-23. P-Flash Protection Scenario Transitions**

| From Protection Scenario | To Protection Scenario[1] | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| 0 | X | X | X | X | | | | |
| 1 | | X | | X | | | | |
| 2 | | | X | X | | | | |
| 3 | | | | X | | | | |
| 4 | | | | X | X | | | |
| 5 | | | X | X | X | X | | |
| 6 | | X | | X | X | | X | |
| 7 | X | X | X | X | X | X | X | X |

1. Allowed transitions marked with X, see Figure 29-14 for a definition of the scenarios.

### 29.3.2.10 EEE Protection Register (EPROT)

The EPROT register defines which buffer RAM EEE partition areas are protected against writes.

Offset Module Base + 0x0009

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | EPOPEN | RNV[6:4] | | | EPDIS | EPS[2:0] | | |
| W | | | | | | | | |
| Reset | F | F | F | F | F | F | F | F |

= Unimplemented or Reserved

**Figure 29-15. EEE Protection Register (EPROT)**

All bits in the EPROT register are readable and writable except for RNV[6:4] which are only readable. The EPOPEN and EPDIS bits can only be written to the protected state. The EPS bits can be written anytime until the EPDIS bit is cleared. If the EPOPEN bit is cleared, the state of the EPDIS and EPS bits is irrelevant.

During the reset sequence, the EPROT register is loaded from the EEE protection byte in the Flash configuration field at global address 0x7F_FF0D located in P-Flash memory (see Table 29-3) as indicated by reset condition F in Figure 29-15. To change the EEE protection that will be loaded during the reset sequence, the P-Flash sector containing the EEE protection byte must be unprotected, then the EEE protection byte must be programmed. If a double bit fault is detected while reading the P-Flash phrase

containing the EEE protection byte during the reset sequence, the EPOPEN bit will be cleared and remaining bits in the EPROT register will be set to leave the buffer RAM EEE partition fully protected.

Trying to write data to any protected area in the buffer RAM EEE partition will result in a protection violation error and the EPVIOLIF flag will be set in the FERSTAT register. Trying to write data to any protected area in the buffer RAM partitioned for user access will not be prevented and the EPVIOLIF flag in the FERSTAT register will not set.

**Table 29-24. EPROT Field Descriptions**

| Field | Description |
|---|---|
| 7<br>EPOPEN | **Enables writes to the Buffer RAM partitioned for EEE**<br>0 The entire buffer RAM EEE partition is protected from writes<br>1 Unprotected buffer RAM EEE partition areas are enabled for writes |
| 6–4<br>RNV[6:4] | **Reserved Nonvolatile Bits** — The RNV bits should remain in the erased state for future enhancements |
| 3<br>EPDIS | **Buffer RAM Protection Address Range Disable** — The EPDIS bit determines whether there is a protected area in a specific region of the buffer RAM EEE partition.<br>0 Protection enabled<br>1 Protection disabled |
| 2–0<br>EPS[2:0] | **Buffer RAM Protection Size** — The EPS[2:0] bits determine the size of the protected area in the buffer RAM EEE partition as shown inTable 29-21. The EPS bits can only be written to while the EPDIS bit is set. |

**Table 29-25. Buffer RAM EEE Partition Protection Address Range**

| EPS[2:0] | Global Address Range | Protected Size |
|---|---|---|
| 000 | 0x13_FFC0 – 0x13_FFFF | 64 bytes |
| 001 | 0x13_FF80 – 0x13_FFFF | 128 bytes |
| 010 | 0x13_FF40 – 0x13_FFFF | 192 bytes |
| 011 | 0x13_FF00 – 0x13_FFFF | 256 bytes |
| 100 | 0x13_FEC0 – 0x13_FFFF | 320 bytes |
| 101 | 0x13_FE80 – 0x13_FFFF | 384 bytes |
| 110 | 0x13_FE40 – 0x13_FFFF | 448 bytes |
| 111 | 0x13_FE00 – 0x13_FFFF | 512 bytes |

### 29.3.2.11 Flash Common Command Object Register (FCCOB)

The FCCOB is an array of six words addressed via the CCOBIX index found in the FCCOBIX register. Byte wide reads and writes are allowed to the FCCOB register.

Offset Module Base + 0x000A

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | CCOB[15:8] | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 29-16. Flash Common Command Object High Register (FCCOBHI)**

Offset Module Base + 0x000B

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | CCOB[7:0] | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 29-17. Flash Common Command Object Low Register (FCCOBLO)**

### 29.3.2.11.1 FCCOB - NVM Command Mode

NVM command mode uses the indexed FCCOB register to provide a command code and its relevant parameters to the Memory Controller. The user first sets up all required FCCOB fields and then initiates the command's execution by writing a 1 to the CCIF bit in the FSTAT register (a 1 written by the user clears the CCIF command completion flag to 0). When the user clears the CCIF bit in the FSTAT register all FCCOB parameter fields are locked and cannot be changed by the user until the command completes (as evidenced by the Memory Controller returning CCIF to 1). Some commands return information to the FCCOB register array.

The generic format for the FCCOB parameter fields in NVM command mode is shown in Table 29-26. The return values are available for reading after the CCIF flag in the FSTAT register has been returned to 1 by the Memory Controller. Writes to the unimplemented parameter fields (CCOBIX = 110 and CCOBIX = 111) are ignored with reads from these fields returning 0x0000.

Table 29-26 shows the generic Flash command format. The high byte of the first word in the CCOB array contains the command code, followed by the parameters for this specific Flash command. For details on the FCCOB settings required by each command, see the Flash command descriptions in Section 29.4.2.

**Table 29-26. FCCOB - NVM Command Mode (Typical Usage)**

| CCOBIX[2:0] | Byte | FCCOB Parameter Fields (NVM Command Mode) |
|---|---|---|
| 000 | HI | FCMD[7:0] defining Flash command |
| | LO | 0, Global address [22:16] |
| 001 | HI | Global address [15:8] |
| | LO | Global address [7:0] |
| 010 | HI | Data 0 [15:8] |
| | LO | Data 0 [7:0] |

**Table 29-26. FCCOB - NVM Command Mode (Typical Usage)**

| CCOBIX[2:0] | Byte | FCCOB Parameter Fields (NVM Command Mode) |
|---|---|---|
| 011 | HI | Data 1 [15:8] |
| | LO | Data 1 [7:0] |
| 100 | HI | Data 2 [15:8] |
| | LO | Data 2 [7:0] |
| 101 | HI | Data 3 [15:8] |
| | LO | Data 3 [7:0] |

### 29.3.2.12 EEE Tag Counter Register (ETAG)

The ETAG register contains the number of outstanding words in the buffer RAM EEE partition that need to be programmed into the D-Flash EEE partition. The ETAG register is decremented prior to the related tagged word being programmed into the D-Flash EEE partition. All tagged words have been programmed into the D-Flash EEE partition once all bits in the ETAG register read 0 and the MGBUSY flag in the FSTAT register reads 0.

Offset Module Base + 0x000C

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | ETAG[15:8] | | | | | | | |
| W | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented or Reserved

**Figure 29-18. EEE Tag Counter High Register (ETAGHI)**

Offset Module Base + 0x000D

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | ETAG[7:0] | | | | | | | |
| W | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented or Reserved

**Figure 29-19. EEE Tag Counter Low Register (ETAGLO)**

All ETAG bits are readable but not writable and are cleared by the Memory Controller.

### 29.3.2.13 Flash ECC Error Results Register (FECCR)

The FECCR registers contain the result of a detected ECC fault for both single bit and double bit faults. The FECCR register provides access to several ECC related fields as defined by the ECCRIX index bits in the FECCRIX register (see Section 29.3.2.4). Once ECC fault information has been stored, no other

fault information will be recorded until the specific ECC fault flag has been cleared. In the event of simultaneous ECC faults, the priority for fault recording is:

1. Double bit fault over single bit fault
2. CPU over XGATE

Offset Module Base + 0x000E

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | ECCR[15:8] | | | | | | | |
| W | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented or Reserved

**Figure 29-20. Flash ECC Error Results High Register (FECCRHI)**

Offset Module Base + 0x000F

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | ECCR[7:0] | | | | | | | |
| W | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented or Reserved

**Figure 29-21. Flash ECC Error Results Low Register (FECCRLO)**

All FECCR bits are readable but not writable.

**Table 29-27. FECCR Index Settings**

| ECCRIX[2:0] | FECCR Register Content | | |
|---|---|---|---|
| | **Bits [15:8]** | **Bit[7]** | **Bits[6:0]** |
| 000 | Parity bits read from Flash block | CPU or XGATE source identity | Global address [22:16] |
| 001 | Global address [15:0] | | |
| 010 | Data 0 [15:0] | | |
| 011 | Data 1 [15:0] (P-Flash only) | | |
| 100 | Data 2 [15:0] (P-Flash only) | | |
| 101 | Data 3 [15:0] (P-Flash only) | | |
| 110 | Not used, returns 0x0000 when read | | |
| 111 | Not used, returns 0x0000 when read | | |

**Table 29-28. FECCR Index=000 Bit Descriptions**

| Field | Description |
|---|---|
| 15:8<br>PAR[7:0] | **ECC Parity Bits** — Contains the 8 parity bits from the 72 bit wide P-Flash data word or the 6 parity bits, allocated to PAR[5:0], from the 22 bit wide D-Flash word with PAR[7:6]=00. |
| 7<br>XBUS01 | **Bus Source Identifier** — The XBUS01 bit determines whether the ECC error was caused by a read access from the CPU or XGATE.<br>0 ECC Error happened on the CPU access<br>1 ECC Error happened on the XGATE access |
| 6–0<br>GADDR[22:16] | **Global Address** — The GADDR[22:16] field contains the upper seven bits of the global address having caused the error. |

The P-Flash word addressed by ECCRIX = 001 contains the lower 16 bits of the global address. The following four words addressed by ECCRIX = 010 to 101 contain the 64-bit wide data phrase. The four data words and the parity byte are the uncorrected data read from the P-Flash block.

The D-Flash word addressed by ECCRIX = 001 contains the lower 16 bits of the global address. The uncorrected 16-bit data word is addressed by ECCRIX = 010.

### 29.3.2.14 Flash Option Register (FOPT)

The FOPT register is the Flash option register.

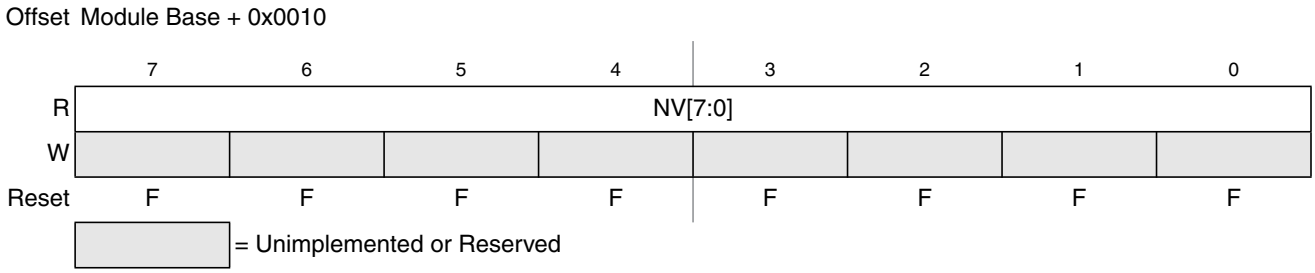



**Figure 29-22. Flash Option Register (FOPT)**

All bits in the FOPT register are readable but are not writable.

During the reset sequence, the FOPT register is loaded from the Flash nonvolatile byte in the Flash configuration field at global address 0x7F_FF0E located in P-Flash memory (see Table 29-3) as indicated by reset condition F in Figure 29-22. If a double bit fault is detected while reading the P-Flash phrase containing the Flash nonvolatile byte during the reset sequence, all bits in the FOPT register will be set.

**Table 29-29. FOPT Field Descriptions**

| Field | Description |
|---|---|
| 7–0<br>NV[7:0] | **Nonvolatile Bits** — The NV[7:0] bits are available as nonvolatile bits. Refer to the device user guide for proper use of the NV bits. |

### 29.3.2.15 Flash Reserved0 Register (FRSV0)

This Flash register is reserved for factory testing.

Offset Module Base + 0x0011

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| W | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented or Reserved

**Figure 29-23. Flash Reserved0 Register (FRSV0)**

All bits in the FRSV0 register read 0 and are not writable.

### 29.3.2.16 Flash Reserved1 Register (FRSV1)

This Flash register is reserved for factory testing.

Offset Module Base + 0x0012

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| W | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented or Reserved

**Figure 29-24. Flash Reserved1 Register (FRSV1)**

All bits in the FRSV1 register read 0 and are not writable.

### 29.3.2.17 Flash Reserved2 Register (FRSV2)

This Flash register is reserved for factory testing.

Offset Module Base + 0x0013

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| W | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented or Reserved

**Figure 29-25. Flash Reserved2 Register (FRSV2)**

All bits in the FRSV2 register read 0 and are not writable.

# 29.4 Functional Description

## 29.4.1 Flash Command Operations

Flash command operations are used to modify Flash memory contents or configure module resources for EEE operation.

The next sections describe:

- How to write the FCLKDIV register that is used to generate a time base (FCLK) derived from OSCCLK for Flash program and erase command operations
- The command write sequence used to set Flash command parameters and launch execution
- Valid Flash commands available for execution

### 29.4.1.1 Writing the FCLKDIV Register

Prior to issuing any Flash program or erase command after a reset, the user is required to write the FCLKDIV register to divide OSCCLK down to a target FCLK of 1 MHz. Table 29-9 shows recommended values for the FDIV field based on OSCCLK frequency.

**NOTE**

Programming or erasing the Flash memory cannot be performed if the bus clock runs at less than 1 MHz. Setting FDIV too high can destroy the Flash memory due to overstress. Setting FDIV too low can result in incomplete programming or erasure of the Flash memory cells.

When the FCLKDIV register is written, the FDIVLD bit is set automatically. If the FDIVLD bit is 0, the FCLKDIV register has not been written since the last reset. If the FCLKDIV register has not been written, any Flash program or erase command loaded during a command write sequence will not execute and the ACCERR bit in the FSTAT register will set.

### 29.4.1.2 Command Write Sequence

The Memory Controller will launch all valid Flash commands entered using a command write sequence.

Before launching a command, the ACCERR and FPVIOL bits in the FSTAT register must be clear (see Section 29.3.2.7) and the CCIF flag should be tested to determine the status of the current command write sequence. If CCIF is 0, the previous command write sequence is still active, a new command write sequence cannot be started, and all writes to the FCCOB register are ignored.

**CAUTION**

Writes to any Flash register must be avoided while a Flash command is active (CCIF=0) to prevent corruption of Flash register contents and Memory Controller behavior.

#### 29.4.1.2.1 Define FCCOB Contents

The FCCOB parameter fields must be loaded with all required parameters for the Flash command being executed. Access to the FCCOB parameter fields is controlled via the CCOBIX bits in the FCCOBIX register (see Section 29.3.2.3).

The contents of the FCCOB parameter fields are transferred to the Memory Controller when the user clears the CCIF command completion flag in the FSTAT register (writing 1 clears the CCIF to 0). The CCIF flag will remain clear until the Flash command has completed. Upon completion, the Memory Controller will return CCIF to 1 and the FCCOB register will be used to communicate any results. The flow for a generic command write sequence is shown in Figure 29-26.

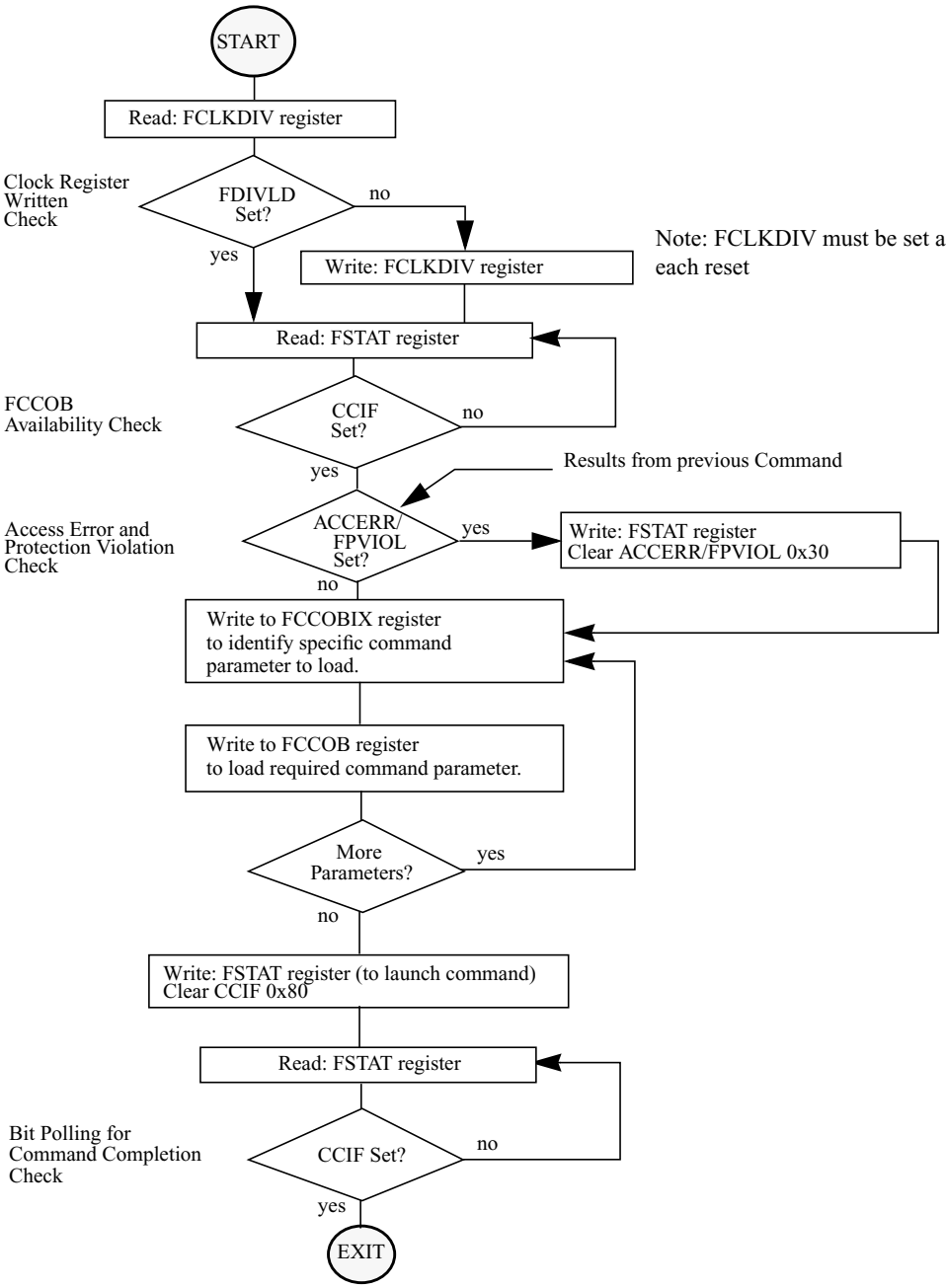



**Figure 29-26. Generic Flash Command Write Sequence Flowchart**

### 29.4.1.3 Valid Flash Module Commands

**Table 29-30. Flash Commands by Mode**

| FCMD | Command | Unsecured | | | | Secured | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | NS (1) | NX (2) | SS(3) | ST(4) | NS (5) | NX (6) | SS(7) | ST(8) |
| 0x01 | Erase Verify All Blocks | * | * | * | * | * | * | * | * |
| 0x02 | Erase Verify Block | * | * | * | * | * | * | * | * |
| 0x03 | Erase Verify P-Flash Section | * | * | * | * | * | | | |
| 0x04 | Read Once | * | * | * | * | * | | | |
| 0x05 | Load Data Field | * | * | * | * | * | | | |
| 0x06 | Program P-Flash | * | * | * | * | * | | | |
| 0x07 | Program Once | * | * | * | * | * | | | |
| 0x08 | Erase All Blocks | | | * | * | | | * | * |
| 0x09 | Erase P-Flash Block | * | * | * | * | * | | | |
| 0x0A | Erase P-Flash Sector | * | * | * | * | * | | | |
| 0x0B | Unsecure Flash | | | * | * | | | * | * |
| 0x0C | Verify Backdoor Access Key | * | | | | * | | | |
| 0x0D | Set User Margin Level | * | * | * | * | * | | | |
| 0x0E | Set Field Margin Level | | | * | * | | | | |
| 0x0F | Full Partition D-Flash | | | * | * | | | | |
| 0x10 | Erase Verify D-Flash Section | * | * | * | * | * | | | |
| 0x11 | Program D-Flash | * | * | * | * | * | | | |
| 0x12 | Erase D-Flash Sector | * | * | * | * | * | | | |
| 0x13 | Enable EEPROM Emulation | * | * | * | * | * | * | * | * |
| 0x14 | Disable EEPROM Emulation | * | * | * | * | * | * | * | * |
| 0x15 | EEPROM Emulation Query | * | * | * | * | * | * | * | * |
| 0x20 | Partition D-Flash | * | * | * | * | * | * | * | * |

1. Unsecured Normal Single Chip mode.
2. Unsecured Normal Expanded mode.
3. Unsecured Special Single Chip mode.
4. Unsecured Special Mode.
5. Secured Normal Single Chip mode.
6. Secured Normal Expanded mode.
7. Secured Special Single Chip mode.
8. Secured Special Mode.

#### 29.4.1.4 P-Flash Commands

Table 29-31 summarizes the valid P-Flash commands along with the effects of the commands on the P-Flash block and other resources within the Flash module.

**Table 29-31. P-Flash Commands**

| FCMD | Command | Function on P-Flash Memory |
|---|---|---|
| 0x01 | Erase Verify All Blocks | Verify that all P-Flash (and D-Flash) blocks are erased. |
| 0x02 | Erase Verify Block | Verify that a P-Flash block is erased. |
| 0x03 | Erase Verify P-Flash Section | Verify that a given number of words starting at the address provided are erased. |
| 0x04 | Read Once | Read a dedicated 64 byte field in the nonvolatile information register in P-Flash block 0 that was previously programmed using the Program Once command. |
| 0x05 | Load Data Field | Load data for simultaneous multiple P-Flash block operations. |
| 0x06 | Program P-Flash | Program a phrase in a P-Flash block and any previously loaded phrases for any other P-Flash block (see Load Data Field command). |
| 0x07 | Program Once | Program a dedicated 64 byte field in the nonvolatile information register in P-Flash block 0 that is allowed to be programmed only once. |
| 0x08 | Erase All Blocks | Erase all P-Flash (and D-Flash) blocks.<br>An erase of all Flash blocks is only possible when the FPLDIS, FPHDIS, and FPOPEN bits in the FPROT register and the EPDIS and EPOPEN bits in the EPROT register are set prior to launching the command. |
| 0x09 | Erase P-Flash Block | Erase a single P-Flash block.<br>An erase of the full P-Flash block is only possible when FPLDIS, FPHDIS and FPOPEN bits in the FPROT register are set prior to launching the command. |
| 0x0A | Erase P-Flash Sector | Erase all bytes in a P-Flash sector. |
| 0x0B | Unsecure Flash | Supports a method of releasing MCU security by erasing all P-Flash (and D-Flash) blocks and verifying that all P-Flash (and D-Flash) blocks are erased. |
| 0x0C | Verify Backdoor Access Key | Supports a method of releasing MCU security by verifying a set of security keys. |
| 0x0D | Set User Margin Level | Specifies a user margin read level for all P-Flash blocks. |
| 0x0E | Set Field Margin Level | Specifies a field margin read level for all P-Flash blocks (special modes only). |

#### 29.4.1.5 D-Flash and EEE Commands

Table 29-32 summarizes the valid D-Flash and EEE commands along with the effects of the commands on the D-Flash block and EEE operation.

**Table 29-32. D-Flash Commands**

| FCMD | Command | Function on D-Flash Memory |
|---|---|---|
| 0x01 | Erase Verify All Blocks | Verify that all D-Flash (and P-Flash) blocks are erased. |
| 0x02 | Erase Verify Block | Verify that the D-Flash block is erased. |

**Table 29-32. D-Flash Commands**

| FCMD | Command | Function on D-Flash Memory |
|---|---|---|
| 0x08 | Erase All Blocks | Erase all D-Flash (and P-Flash) blocks.<br>An erase of all Flash blocks is only possible when the FPLDIS, FPHDIS, and FPOPEN bits in the FPROT register and the EPDIS and EPOPEN bits in the EPROT register are set prior to launching the command. |
| 0x0B | Unsecure Flash | Supports a method of releasing MCU security by erasing all D-Flash (and P-Flash) blocks and verifying that all D-Flash (and P-Flash) blocks are erased. |
| 0x0D | Set User Margin Level | Specifies a user margin read level for the D-Flash block. |
| 0x0E | Set Field Margin Level | Specifies a field margin read level for the D-Flash block (special modes only). |
| 0x0F | Full Partition D-Flash | Erase the D-Flash block and partition an area of the D-Flash block for user access. |
| 0x10 | Erase Verify D-Flash Section | Verify that a given number of words starting at the address provided are erased. |
| 0x11 | Program D-Flash | Program up to four words in the D-Flash block. |
| 0x12 | Erase D-Flash Sector | Erase all bytes in a sector of the D-Flash block. |
| 0x13 | Enable EEPROM Emulation | Enable EEPROM emulation where writes to the buffer RAM EEE partition will be copied to the D-Flash EEE partition. |
| 0x14 | Disable EEPROM Emulation | Suspend all current erase and program activity related to EEPROM emulation but leave current EEE tags set. |
| 0x15 | EEPROM Emulation Query | Returns EEE partition and status variables. |
| 0x20 | Partition D-Flash | Partition an area of the D-Flash block for user access. |

## 29.4.2 Flash Command Description

This section provides details of all available Flash commands launched by a command write sequence. The ACCERR bit in the FSTAT register will be set during the command write sequence if any of the following illegal steps are performed, causing the command not to be processed by the Memory Controller:

- Starting any command write sequence that programs or erases Flash memory before initializing the FCLKDIV register
- Writing an invalid command as part of the command write sequence
- For additional possible errors, refer to the error handling table provided for each command

If a Flash block is read during execution of an algorithm (CCIF = 0) on that same block, the read operation will return invalid data. If the SFDIF or DFDIF flags were not previously set when the invalid read operation occurred, both the SFDIF and DFDIF flags will be set and the FECCR registers will be loaded with the global address used in the invalid read operation with the data and parity fields set to all 0.

If the ACCERR or FPVIOL bits are set in the FSTAT register, the user must clear these bits before starting any command write sequence (see Section 29.3.2.7).

**CAUTION**

A Flash word or phrase must be in the erased state before being programmed. Cumulative programming of bits within a Flash word or phrase is not allowed.

### 29.4.2.1 Erase Verify All Blocks Command

The Erase Verify All Blocks command will verify that all P-Flash and D-Flash blocks have been erased.

**Table 29-33. Erase Verify All Blocks Command FCCOB Requirements**

| CCOBIX[2:0] | FCCOB Parameters | |
|---|---|---|
| 000 | 0x01 | Not required |

Upon clearing CCIF to launch the Erase Verify All Blocks command, the Memory Controller will verify that the entire Flash memory space is erased. The CCIF flag will set after the Erase Verify All Blocks operation has completed.

**Table 29-34. Erase Verify All Blocks Command Error Handling**

| Register | Error Bit | Error Condition |
|---|---|---|
| FSTAT | ACCERR | Set if CCOBIX[2:0] != 000 at command launch |
| | | Set if a Load Data Field command sequence is currently active |
| | FPVIOL | None |
| | MGSTAT1 | Set if any errors have been encountered during the read |
| | MGSTAT0 | Set if any non-correctable errors have been encountered during the read |
| FERSTAT | EPVIOLIF | None |

### 29.4.2.2 Erase Verify Block Command

The Erase Verify Block command allows the user to verify that an entire P-Flash or D-Flash block has been erased. The FCCOB upper global address bits determine which block must be verified.

**Table 29-35. Erase Verify Block Command FCCOB Requirements**

| CCOBIX[2:0] | FCCOB Parameters | |
|---|---|---|
| 000 | 0x02 | Global address [22:16] of the Flash block to be verified. |

Upon clearing CCIF to launch the Erase Verify Block command, the Memory Controller will verify that the selected P-Flash or D-Flash block is erased. The CCIF flag will set after the Erase Verify Block operation has completed.

**Table 29-36. Erase Verify Block Command Error Handling**

| Register | Error Bit | Error Condition |
|---|---|---|
| FSTAT | ACCERR | Set if CCOBIX[2:0] != 000 at command launch |
| | | Set if a Load Data Field command sequence is currently active |
| | | Set if an invalid global address [22:16] is supplied |
| | FPVIOL | None |
| | MGSTAT1 | Set if any errors have been encountered during the read |
| | MGSTAT0 | Set if any non-correctable errors have been encountered during the read |
| FERSTAT | EPVIOLIF | None |

### 29.4.2.3 Erase Verify P-Flash Section Command

The Erase Verify P-Flash Section command will verify that a section of code in the P-Flash memory is erased. The Erase Verify P-Flash Section command defines the starting point of the code to be verified and the number of phrases. The section to be verified cannot cross a 256 Kbyte boundary in the P-Flash memory space.

**Table 29-37. Erase Verify P-Flash Section Command FCCOB Requirements**

| CCOBIX[2:0] | FCCOB Parameters | |
|---|---|---|
| 000 | 0x03 | Global address [22:16] of a P-Flash block |
| 001 | Global address [15:0] of the first phrase to be verified | |
| 010 | Number of phrases to be verified | |

Upon clearing CCIF to launch the Erase Verify P-Flash Section command, the Memory Controller will verify the selected section of Flash memory is erased. The CCIF flag will set after the Erase Verify P-Flash Section operation has completed.

**Table 29-38. Erase Verify P-Flash Section Command Error Handling**

| Register | Error Bit | Error Condition |
|---|---|---|
| FSTAT | ACCERR | Set if CCOBIX[2:0] != 010 at command launch |
| | | Set if a Load Data Field command sequence is currently active |
| | | Set if command not available in current mode (see Table 29-30) |
| | | Set if an invalid global address [22:0] is supplied |
| | | Set if a misaligned phrase address is supplied (global address [2:0] != 000) |
| | | Set if the requested section crosses a 256 Kbyte boundary |
| | FPVIOL | None |
| | MGSTAT1 | Set if any errors have been encountered during the read |
| | MGSTAT0 | Set if any non-correctable errors have been encountered during the read |

**Table 29-38. Erase Verify P-Flash Section Command Error Handling**

| Register | Error Bit | Error Condition |
|---|---|---|
| FERSTAT | EPVIOLIF | None |

### 29.4.2.4 Read Once Command

The Read Once command provides read access to a reserved 64 byte field (8 phrases) located in the nonvolatile information register of P-Flash block 0. The Read Once field is programmed using the Program Once command described in Section 29.4.2.7. The Read Once command must not be executed from the Flash block containing the Program Once reserved field to avoid code runaway.

**Table 29-39. Read Once Command FCCOB Requirements**

| CCOBIX[2:0] | FCCOB Parameters | |
|---|---|---|
| 000 | 0x04 | Not Required |
| 001 | Read Once phrase index (0x0000 - 0x0007) | |
| 010 | Read Once word 0 value | |
| 011 | Read Once word 1 value | |
| 100 | Read Once word 2 value | |
| 101 | Read Once word 3 value | |

Upon clearing CCIF to launch the Read Once command, a Read Once phrase is fetched and stored in the FCCOB indexed register. The CCIF flag will set after the Read Once operation has completed. Valid phrase index values for the Read Once command range from 0x0000 to 0x0007. During execution of the Read Once command, any attempt to read addresses within P-Flash block 0 will return invalid data.

**Table 29-40. Read Once Command Error Handling**

| Register | Error Bit | Error Condition |
|---|---|---|
| FSTAT | ACCERR | Set if CCOBIX[2:0] != 001 at command launch |
| | | Set if a Load Data Field command sequence is currently active |
| | | Set if command not available in current mode (see Table 29-30) |
| | | Set if an invalid phrase index is supplied |
| | FPVIOL | None |
| | MGSTAT1 | Set if any errors have been encountered during the read |
| | MGSTAT0 | Set if any non-correctable errors have been encountered during the read |
| FERSTAT | EPVIOLIF | None |

### 29.4.2.5 Load Data Field Command

The Load Data Field command is executed to provide FCCOB parameters for multiple P-Flash blocks for a future simultaneous program operation in the P-Flash memory space.

**Table 29-41. Load Data Field Command FCCOB Requirements**

| CCOBIX[2:0] | FCCOB Parameters | |
|---|---|---|
| 000 | 0x05 | Global address [22:16] to identify P-Flash block |
| 001 | Global address [15:0] of phrase location to be programmed(1) | |
| 010 | Word 0 | |
| 011 | Word 1 | |
| 100 | Word 2 | |
| 101 | Word 3 | |

1. Global address [2:0] must be 000

Upon clearing CCIF to launch the Load Data Field command, the FCCOB registers will be transferred to the Memory Controller and be programmed in the block specified at the global address given with a future Program P-Flash command launched on a P-Flash block. The CCIF flag will set after the Load Data Field operation has completed. Note that once a Load Data Field command sequence has been initiated, the Load Data Field command sequence will be cancelled if any command other than Load Data Field or the future Program P-Flash is launched. Similarly, if an error occurs after launching a Load Data Field or Program P-Flash command, the associated Load Data Field command sequence will be cancelled.

**Table 29-42. Load Data Field Command Error Handling**

| Register | Error Bit | Error Condition |
|---|---|---|
| FSTAT | ACCERR | Set if CCOBIX[2:0] != 101 at command launch |
| | | Set if command not available in current mode (see Table 29-30) |
| | | Set if an invalid global address [22:0] is supplied |
| | | Set if a misaligned phrase address is supplied (global address [2:0] != 000) |
| | | Set if a Load Data Field command sequence is currently active and the selected block has previously been selected in the same command sequence |
| | | Set if a Load Data Field command sequence is currently active and global address [17:0] does not match that previously supplied in the same command sequence |
| | FPVIOL | Set if the global address [22:0] points to a protected area |
| | MGSTAT1 | None |
| | MGSTAT0 | None |
| FERSTAT | EPVIOLIF | None |

### 29.4.2.6 Program P-Flash Command

The Program P-Flash operation will program a previously erased phrase in the P-Flash memory using an embedded algorithm.

**CAUTION**

A P-Flash phrase must be in the erased state before being programmed. Cumulative programming of bits within a Flash phrase is not allowed.

**Table 29-43. Program P-Flash Command FCCOB Requirements**

| CCOBIX[2:0] | FCCOB Parameters | |
|---|---|---|
| 000 | 0x06 | Global address [22:16] to identify P-Flash block |
| 001 | Global address [15:0] of phrase location to be programmed(1) | |
| 010 | Word 0 program value | |
| 011 | Word 1 program value | |
| 100 | Word 2 program value | |
| 101 | Word 3 program value | |

1. Global address [2:0] must be 000

Upon clearing CCIF to launch the Program P-Flash command, the Memory Controller will program the data words to the supplied global address and will then proceed to verify the data words read back as expected. The CCIF flag will set after the Program P-Flash operation has completed.

**Table 29-44. Program P-Flash Command Error Handling**

| Register | Error Bit | Error Condition |
|---|---|---|
| FSTAT | ACCERR | Set if CCOBIX[2:0] != 101 at command launch |
| | | Set if command not available in current mode (see Table 29-30) |
| | | Set if an invalid global address [22:0] is supplied |
| | | Set if a misaligned phrase address is supplied (global address [2:0] != 000) |
| | | Set if a Load Data Field command sequence is currently active and the selected block has previously been selected in the same command sequence |
| | | Set if a Load Data Field command sequence is currently active and global address [17:0] does not match that previously supplied in the same command sequence |
| | FPVIOL | Set if the global address [22:0] points to a protected area |
| | MGSTAT1 | Set if any errors have been encountered during the verify operation |
| | MGSTAT0 | Set if any non-correctable errors have been encountered during the verify operation |
| FERSTAT | EPVIOLIF | None |

### 29.4.2.7 Program Once Command

The Program Once command restricts programming to a reserved 64 byte field (8 phrases) in the nonvolatile information register located in P-Flash block 0. The Program Once reserved field can be read using the Read Once command as described in Section 29.4.2.4. The Program Once command must only be issued once since the nonvolatile information register in P-Flash block 0 cannot be erased. The Program

Once command must not be executed from the Flash block containing the Program Once reserved field to avoid code runaway.

**Table 29-45. Program Once Command FCCOB Requirements**

| CCOBIX[2:0] | FCCOB Parameters | |
|---|---|---|
| 000 | 0x07 | Not Required |
| 001 | Program Once phrase index (0x0000 - 0x0007) | |
| 010 | Program Once word 0 value | |
| 011 | Program Once word 1 value | |
| 100 | Program Once word 2 value | |
| 101 | Program Once word 3 value | |

Upon clearing CCIF to launch the Program Once command, the Memory Controller first verifies that the selected phrase is erased. If erased, then the selected phrase will be programmed and then verified with read back. The CCIF flag will remain clear, setting only after the Program Once operation has completed.

The reserved nonvolatile information register accessed by the Program Once command cannot be erased and any attempt to program one of these phrases a second time will not be allowed. Valid phrase index values for the Program Once command range from 0x0000 to 0x0007. During execution of the Program Once command, any attempt to read addresses within P-Flash block 0 will return invalid data.

**Table 29-46. Program Once Command Error Handling**

| Register | Error Bit | Error Condition |
|---|---|---|
| FSTAT | ACCERR | Set if CCOBIX[2:0] != 101 at command launch |
| | | Set if a Load Data Field command sequence is currently active |
| | | Set if command not available in current mode (see Table 29-30) |
| | | Set if an invalid phrase index is supplied |
| | | Set if the requested phrase has already been programmed(1) |
| | FPVIOL | None |
| | MGSTAT1 | Set if any errors have been encountered during the verify operation |
| | MGSTAT0 | Set if any non-correctable errors have been encountered during the verify operation |
| FERSTAT | EPVIOLIF | None |

1. If a Program Once phrase is initially programmed to 0xFFFF_FFFF_FFFF_FFFF, the Program Once command will be allowed to execute again on that same phrase.

### 29.4.2.8 Erase All Blocks Command

The Erase All Blocks operation will erase the entire P-Flash and D-Flash memory space including the EEE nonvolatile information register.

**Table 29-47. Erase All Blocks Command FCCOB Requirements**

| CCOBIX[2:0] | FCCOB Parameters | |
|---|---|---|
| 000 | 0x08 | Not required |

Upon clearing CCIF to launch the Erase All Blocks command, the Memory Controller will erase the entire Flash memory space and verify that it is erased. If the Memory Controller verifies that the entire Flash memory space was properly erased, security will be released. During the execution of this command (CCIF=0) the user must not write to any Flash module register. The CCIF flag will set after the Erase All Blocks operation has completed.

**Table 29-48. Erase All Blocks Command Error Handling**

| Register | Error Bit | Error Condition |
|---|---|---|
| FSTAT | ACCERR | Set if CCOBIX[2:0] != 000 at command launch |
| | | Set if a Load Data Field command sequence is currently active |
| | | Set if command not available in current mode (see Table 29-30) |
| | FPVIOL | Set if any area of the P-Flash memory is protected |
| | MGSTAT1 | Set if any errors have been encountered during the verify operation |
| | MGSTAT0 | Set if any non-correctable errors have been encountered during the verify operation |
| FERSTAT | EPVIOLIF | Set if any area of the buffer RAM EEE partition is protected |

### 29.4.2.9 Erase P-Flash Block Command

The Erase P-Flash Block operation will erase all addresses in a P-Flash block.

**Table 29-49. Erase P-Flash Block Command FCCOB Requirements**

| CCOBIX[2:0] | FCCOB Parameters | |
|---|---|---|
| 000 | 0x09 | Global address [22:16] to identify P-Flash block |
| 001 | Global address [15:0] in P-Flash block to be erased | |

Upon clearing CCIF to launch the Erase P-Flash Block command, the Memory Controller will erase the selected P-Flash block and verify that it is erased. The CCIF flag will set after the Erase P-Flash Block operation has completed.

**Table 29-50. Erase P-Flash Block Command Error Handling**

| Register | Error Bit | Error Condition |
|---|---|---|
| FSTAT | ACCERR | Set if CCOBIX[2:0] != 001 at command launch |
| | | Set if a Load Data Field command sequence is currently active |
| | | Set if command not available in current mode (see Table 29-30) |
| | | Set if an invalid global address [22:16] is supplied |
| | FPVIOL | Set if an area of the selected P-Flash block is protected |
| | MGSTAT1 | Set if any errors have been encountered during the verify operation |
| | MGSTAT0 | Set if any non-correctable errors have been encountered during the verify operation |
| FERSTAT | EPVIOLIF | None |

### 29.4.2.10 Erase P-Flash Sector Command

The Erase P-Flash Sector operation will erase all addresses in a P-Flash sector.

**Table 29-51. Erase P-Flash Sector Command FCCOB Requirements**

| CCOBIX[2:0] | FCCOB Parameters | |
|---|---|---|
| 000 | 0x0A | Global address [22:16] to identify P-Flash block to be erased |
| 001 | Global address [15:0] anywhere within the sector to be erased. Refer to Section 29.1.2.1 for the P-Flash sector size. | |

Upon clearing CCIF to launch the Erase P-Flash Sector command, the Memory Controller will erase the selected Flash sector and then verify that it is erased. The CCIF flag will be set after the Erase P-Flash Sector operation has completed.

**Table 29-52. Erase P-Flash Sector Command Error Handling**

| Register | Error Bit | Error Condition |
|---|---|---|
| FSTAT | ACCERR | Set if CCOBIX[2:0] != 001 at command launch |
| | | Set if a Load Data Field command sequence is currently active |
| | | Set if command not available in current mode (see Table 29-30) |
| | | Set if an invalid global address [22:16] is supplied |
| | | Set if a misaligned phrase address is supplied (global address [2:0] != 000) |
| | FPVIOL | Set if the selected P-Flash sector is protected |
| | MGSTAT1 | Set if any errors have been encountered during the verify operation |
| | MGSTAT0 | Set if any non-correctable errors have been encountered during the verify operation |
| FERSTAT | EPVIOLIF | None |

### 29.4.2.11 Unsecure Flash Command

The Unsecure Flash command will erase the entire P-Flash and D-Flash memory space and, if the erase is successful, will release security.

**Table 29-53. Unsecure Flash Command FCCOB Requirements**

| CCOBIX[2:0] | FCCOB Parameters | |
|---|---|---|
| 000 | 0x0B | Not required |

Upon clearing CCIF to launch the Unsecure Flash command, the Memory Controller will erase the entire P-Flash and D-Flash memory space and verify that it is erased. If the Memory Controller verifies that the entire Flash memory space was properly erased, security will be released. If the erase verify is not successful, the Unsecure Flash operation sets MGSTAT1 and terminates without changing the security state. During the execution of this command (CCIF=0) the user must not write to any Flash module register. The CCIF flag is set after the Unsecure Flash operation has completed.

**Table 29-54. Unsecure Flash Command Error Handling**

| Register | Error Bit | Error Condition |
|---|---|---|
| FSTAT | ACCERR | Set if CCOBIX[2:0] != 000 at command launch |
| | | Set if a Load Data Field command sequence is currently active |
| | | Set if command not available in current mode (see Table 29-30) |
| | FPVIOL | Set if any area of the P-Flash memory is protected |
| | MGSTAT1 | Set if any errors have been encountered during the verify operation |
| | MGSTAT0 | Set if any non-correctable errors have been encountered during the verify operation |
| FERSTAT | EPVIOLIF | Set if any area of the buffer RAM EEE partition is protected |

### 29.4.2.12 Verify Backdoor Access Key Command

The Verify Backdoor Access Key command will only execute if it is enabled by the KEYEN bits in the FSEC register (see Table 29-11). The Verify Backdoor Access Key command releases security if user-supplied keys match those stored in the Flash security bytes of the Flash configuration field (see Table 29-3). The Verify Backdoor Access Key command must not be executed from the Flash block containing the backdoor comparison key to avoid code runaway.

**Table 29-55. Verify Backdoor Access Key Command FCCOB Requirements**

| CCOBIX[2:0] | FCCOB Parameters | |
|---|---|---|
| 000 | 0x0C | Not required |
| 001 | Key 0 | |
| 010 | Key 1 | |
| 011 | Key 2 | |
| 100 | Key 3 | |

Upon clearing CCIF to launch the Verify Backdoor Access Key command, the Memory Controller will check the FSEC KEYEN bits to verify that this command is enabled. If not enabled, the Memory Controller sets the ACCERR bit in the FSTAT register and terminates. If the command is enabled, the Memory Controller compares the key provided in FCCOB to the backdoor comparison key in the Flash configuration field with Key 0 compared to 0x7F_FF00, etc. If the backdoor keys match, security will be released. If the backdoor keys do not match, security is not released and all future attempts to execute the Verify Backdoor Access Key command are aborted (set ACCERR) until a reset occurs. The CCIF flag is set after the Verify Backdoor Access Key operation has completed.

**Table 29-56. Verify Backdoor Access Key Command Error Handling**

| Register | Error Bit | Error Condition |
|---|---|---|
| FSTAT | ACCERR | Set if CCOBIX[2:0] != 100 at command launch |
| | | Set if a Load Data Field command sequence is currently active |
| | | Set if an incorrect backdoor key is supplied |
| | | Set if backdoor key access has not been enabled (KEYEN[1:0] != 10, see Section 29.3.2.2) |
| | | Set if the backdoor key has mismatched since the last reset |
| | FPVIOL | None |
| | MGSTAT1 | None |
| | MGSTAT0 | None |
| FERSTAT | EPVIOLIF | None |

### 29.4.2.13 Set User Margin Level Command

The Set User Margin Level command causes the Memory Controller to set the margin level for future read operations of a specific P-Flash or D-Flash block.

**Table 29-57. Set User Margin Level Command FCCOB Requirements**

| CCOBIX[2:0] | FCCOB Parameters | |
|---|---|---|
| 000 | 0x0D | Global address [22:16] to identify the Flash block |
| 001 | Margin level setting | |

Upon clearing CCIF to launch the Set User Margin Level command, the Memory Controller will set the user margin level for the targeted block and then set the CCIF flag.

Valid margin level settings for the Set User Margin Level command are defined in Table 29-58.

**Table 29-58. Valid Set User Margin Level Settings**

| CCOB (CCOBIX=001) | Level Description |
|---|---|
| 0x0000 | Return to Normal Level |
| 0x0001 | User Margin-1 Level(1) |

**Table 29-58. Valid Set User Margin Level Settings**

| CCOB (CCOBIX=001) | Level Description |
|---|---|
| 0x0002 | User Margin-0 Level(2) |

1. Read margin to the erased state
2. Read margin to the programmed state

**Table 29-59. Set User Margin Level Command Error Handling**

| Register | Error Bit | Error Condition |
|---|---|---|
| FSTAT | ACCERR | Set if CCOBIX[2:0] != 001 at command launch |
| | | Set if a Load Data Field command sequence is currently active |
| | | Set if command not available in current mode (see Table 29-30) |
| | | Set if an invalid global address [22:16] is supplied |
| | | Set if an invalid margin level setting is supplied |
| | FPVIOL | None |
| | MGSTAT1 | None |
| | MGSTAT0 | None |
| FERSTAT | EPVIOLIF | None |

**NOTE**

User margin levels can be used to check that Flash memory contents have adequate margin for normal level read operations. If unexpected results are encountered when checking Flash memory contents at user margin levels, a potential loss of information has been detected.

### 29.4.2.14 Set Field Margin Level Command

The Set Field Margin Level command, valid in special modes only, causes the Memory Controller to set the margin level specified for future read operations of a specific P-Flash or D-Flash block.

**Table 29-60. Set Field Margin Level Command FCCOB Requirements**

| CCOBIX[2:0] | FCCOB Parameters | |
|---|---|---|
| 000 | 0x0E | Global address [22:16] to identify the Flash block |
| 001 | Margin level setting | |

Upon clearing CCIF to launch the Set Field Margin Level command, the Memory Controller will set the field margin level for the targeted block and then set the CCIF flag.

Valid margin level settings for the Set Field Margin Level command are defined in Table 29-61.

**Table 29-61. Valid Set Field Margin Level Settings**

| CCOB (CCOBIX=001) | Level Description |
|---|---|
| 0x0000 | Return to Normal Level |
| 0x0001 | User Margin-1 Level(1) |
| 0x0002 | User Margin-0 Level(2) |
| 0x0003 | Field Margin-1 Level[1] |
| 0x0004 | Field Margin-0 Level[2] |

1. Read margin to the erased state
2. Read margin to the programmed state

**Table 29-62. Set Field Margin Level Command Error Handling**

| Register | Error Bit | Error Condition |
|---|---|---|
| FSTAT | ACCERR | Set if CCOBIX[2:0] != 001 at command launch |
| | | Set if a Load Data Field command sequence is currently active |
| | | Set if command not available in current mode (see Table 29-30) |
| | | Set if an invalid global address [22:16] is supplied |
| | | Set if an invalid margin level setting is supplied |
| | FPVIOL | None |
| | MGSTAT1 | None |
| | MGSTAT0 | None |
| FERSTAT | EPVIOLIF | None |

**CAUTION**

Field margin levels must only be used during verify of the initial factory programming.

**NOTE**

Field margin levels can be used to check that Flash memory contents have adequate margin for data retention at the normal level setting. If unexpected results are encountered when checking Flash memory contents at field margin levels, the Flash memory contents should be erased and reprogrammed.

### 29.4.2.15 Full Partition D-Flash Command

The Full Partition D-Flash command allows the user to allocate sectors within the D-Flash block for applications and a partition within the buffer RAM for EEPROM access. The D-Flash block consists of 128 sectors with 256 bytes per sector.

**Table 29-63. Full Partition D-Flash Command FCCOB Requirements**

| CCOBIX[2:0] | FCCOB Parameters | |
|---|---|---|
| 000 | 0x0F | Not required |
| 001 | Number of 256 byte sectors for the D-Flash user partition (DFPART) | |
| 010 | Number of 256 byte sectors for buffer RAM EEE partition (ERPART) | |

Upon clearing CCIF to launch the Full Partition D-Flash command, the following actions are taken to define a partition within the D-Flash block for direct access (DFPART) and a partition within the buffer RAM for EEE use (ERPART):

- Validate the DFPART and ERPART values provided:
  - DFPART <= 128 (maximum number of 256 byte sectors in D-Flash block)
  - ERPART <= 16 (maximum number of 256 byte sectors in buffer RAM)
  - If ERPART > 0, 128 - DFPART >= 12 (minimum number of 256 byte sectors in the D-Flash block required to support EEE)
  - If ERPART > 0, ((128-DFPART)/ERPART) >= 8 (minimum ratio of D-Flash EEE space to buffer RAM EEE space to support EEE)
- Erase the D-Flash block and the EEE nonvolatile information register
- Program DFPART to the EEE nonvolatile information register at global address 0x12_0000 (see Table 29-7)
- Program a duplicate DFPART to the EEE nonvolatile information register at global address 0x12_0002 (see Table 29-7)
- Program ERPART to the EEE nonvolatile information register at global address 0x12_0004 (see Table 29-7)
- Program a duplicate ERPART to the EEE nonvolatile information register at global address 0x12_0006 (see Table 29-7)

The D-Flash user partition will start at global address 0x10_0000. The buffer RAM EEE partition will end at global address 0x13_FFFF. After the Full Partition D-Flash operation has completed, the CCIF flag will set.

Running the Full Partition D-Flash command a second time will result in the previous partition values and the entire D-Flash memory being erased. The data value written corresponds to the number of 256 byte sectors allocated for either direct D-Flash access (DFPART) or buffer RAM EEE access (ERPART).

**Table 29-64. Full Partition D-Flash Command Error Handling**

| Register | Error Bit | Error Condition |
|---|---|---|
| FSTAT | ACCERR | Set if CCOBIX[2:0] != 010 at command launch |
| | | Set if a Load Data Field command sequence is currently active |
| | | Set if command not available in current mode (see Table 29-30) |
| | | Set if an invalid DFPART or ERPART selection is supplied |
| | FPVIOL | None |
| | MGSTAT1 | Set if any errors have been encountered during the read |
| | MGSTAT0 | Set if any non-correctable errors have been encountered during the read |
| FERSTAT | EPVIOLIF | None |

### 29.4.2.16 Erase Verify D-Flash Section Command

The Erase Verify D-Flash Section command will verify that a section of code in the D-Flash user partition is erased. The Erase Verify D-Flash Section command defines the starting point of the data to be verified and the number of words.

**Table 29-65. Erase Verify D-Flash Section Command FCCOB Requirements**

| CCOBIX[2:0] | FCCOB Parameters | |
|---|---|---|
| 000 | 0x10 | Global address [22:16] to identify the D-Flash block |
| 001 | Global address [15:0] of the first word to be verified | |
| 010 | Number of words to be verified | |

Upon clearing CCIF to launch the Erase Verify D-Flash Section command, the Memory Controller will verify the selected section of D-Flash memory is erased. The CCIF flag will set after the Erase Verify D-Flash Section operation has completed.

**Table 29-66. Erase Verify D-Flash Section Command Error Handling**

| Register | Error Bit | Error Condition |
|---|---|---|
| FSTAT | ACCERR | Set if CCOBIX[2:0] != 010 at command launch |
| | | Set if a Load Data Field command sequence is currently active |
| | | Set if command not available in current mode (see Table 29-30) |
| | | Set if an invalid global address [22:0] is supplied |
| | | Set if a misaligned word address is supplied (global address [0] != 0) |
| | | Set if the global address [22:0] points to an area of the D-Flash EEE partition |
| | | Set if the requested section breaches the end of the D-Flash block or goes into the D-Flash EEE partition |
| | FPVIOL | None |
| | MGSTAT1 | Set if any errors have been encountered during the read |
| | MGSTAT0 | Set if any non-correctable errors have been encountered during the read |
| FERSTAT | EPVIOLIF | None |

### 29.4.2.17 Program D-Flash Command

The Program D-Flash operation programs one to four previously erased words in the D-Flash user partition. The Program D-Flash operation will confirm that the targeted location(s) were successfully programmed upon completion.

**CAUTION**

A Flash word must be in the erased state before being programmed. Cumulative programming of bits within a Flash word is not allowed.

**Table 29-67. Program D-Flash Command FCCOB Requirements**

| CCOBIX[2:0] | FCCOB Parameters | |
|---|---|---|
| 000 | 0x11 | Global address [22:16] to identify the D-Flash block |
| 001 | Global address [15:0] of word to be programmed | |
| 010 | Word 0 program value | |
| 011 | Word 1 program value, if desired | |
| 100 | Word 2 program value, if desired | |
| 101 | Word 3 program value, if desired | |

Upon clearing CCIF to launch the Program D-Flash command, the user-supplied words will be transferred to the Memory Controller and be programmed. The CCOBIX index value at Program D-Flash command launch determines how many words will be programmed in the D-Flash block. No protection checks are made in the Program D-Flash operation on the D-Flash block, only access error checks. The CCIF flag is set when the operation has completed.

**Table 29-68. Program D-Flash Command Error Handling**

| Register | Error Bit | Error Condition |
|---|---|---|
| FSTAT | ACCERR | Set if CCOBIX[2:0] < 010 at command launch |
| | | Set if CCOBIX[2:0] > 101 at command launch |
| | | Set if a Load Data Field command sequence is currently active |
| | | Set if command not available in current mode (see Table 29-30) |
| | | Set if an invalid global address [22:0] is supplied |
| | | Set if a misaligned word address is supplied (global address [0] != 0) |
| | | Set if the global address [22:0] points to an area in the D-Flash EEE partition |
| | | Set if the requested group of words breaches the end of the D-Flash block or goes into the D-Flash EEE partition |
| | FPVIOL | None |
| | MGSTAT1 | Set if any errors have been encountered during the verify operation |
| | MGSTAT0 | Set if any non-correctable errors have been encountered during the verify operation |
| FERSTAT | EPVIOLIF | None |

### 29.4.2.18 Erase D-Flash Sector Command

The Erase D-Flash Sector operation will erase all addresses in a sector of the D-Flash user partition.

**Table 29-69. Erase D-Flash Sector Command FCCOB Requirements**

| CCOBIX[2:0] | FCCOB Parameters | |
|---|---|---|
| 000 | 0x12 | Global address [22:16] to identify D-Flash block |
| 001 | Global address [15:0] anywhere within the sector to be erased. See Section 29.1.2.2 for D-Flash sector size. | |

Upon clearing CCIF to launch the Erase D-Flash Sector command, the Memory Controller will erase the selected Flash sector and verify that it is erased. The CCIF flag will set after the Erase D-Flash Sector operation has completed.

**Table 29-70. Erase D-Flash Sector Command Error Handling**

| Register | Error Bit | Error Condition |
|---|---|---|
| FSTAT | ACCERR | Set if CCOBIX[2:0] != 001 at command launch |
| | | Set if a Load Data Field command sequence is currently active |
| | | Set if command not available in current mode (see Table 29-30) |
| | | Set if an invalid global address [22:0] is supplied |
| | | Set if a misaligned word address is supplied (global address [0] != 0) |
| | | Set if the global address [22:0] points to the D-Flash EEE partition |
| | FPVIOL | None |
| | MGSTAT1 | Set if any errors have been encountered during the verify operation |
| | MGSTAT0 | Set if any non-correctable errors have been encountered during the verify operation |
| FERSTAT | EPVIOLIF | None |

### 29.4.2.19 Enable EEPROM Emulation Command

The Enable EEPROM Emulation command causes the Memory Controller to enable EEE activity. EEE activity is disabled after any reset.

**Table 29-71. Enable EEPROM Emulation Command FCCOB Requirements**

| CCOBIX[2:0] | FCCOB Parameters | |
|---|---|---|
| 000 | 0x13 | Not required |

Upon clearing CCIF to launch the Enable EEPROM Emulation command, the CCIF flag will set after the Memory Controller enables EEE operations using the contents of the EEE tag RAM and tag counter. The Full Partition D-Flash or the Partition D-Flash command must be run prior to launching the Enable EEPROM Emulation command.

**Table 29-72. Enable EEPROM Emulation Command Error Handling**

| Register | Error Bit | Error Condition |
|---|---|---|
| FSTAT | ACCERR | Set if CCOBIX[2:0] != 000 at command launch |
| | | Set if a Load Data Field command sequence is currently active |
| | | Set if Full Partition D-Flash or Partition D-Flash command not previously run |
| | FPVIOL | None |
| | MGSTAT1 | None |
| | MGSTAT0 | None |
| FERSTAT | EPVIOLIF | None |

### 29.4.2.20 Disable EEPROM Emulation Command

The Disable EEPROM Emulation command causes the Memory Controller to suspend current EEE activity.

**Table 29-73. Disable EEPROM Emulation Command FCCOB Requirements**

| CCOBIX[2:0] | FCCOB Parameters | |
|---|---|---|
| 000 | 0x14 | Not required |

Upon clearing CCIF to launch the Disable EEPROM Emulation command, the Memory Controller will halt EEE operations at the next convenient point without clearing the EEE tag RAM or tag counter before setting the CCIF flag.

**Table 29-74. Disable EEPROM Emulation Command Error Handling**

| Register | Error Bit | Error Condition |
|---|---|---|
| FSTAT | ACCERR | Set if CCOBIX[2:0] != 000 at command launch |
| | | Set if a Load Data Field command sequence is currently active |
| | | Set if Full Partition D-Flash or Partition D-Flash command not previously run |
| | FPVIOL | None |
| | MGSTAT1 | None |
| | MGSTAT0 | None |
| FERSTAT | EPVIOLIF | None |

### 29.4.2.21 EEPROM Emulation Query Command

The EEPROM Emulation Query command returns EEE partition and status variables.

**Table 29-75. EEPROM Emulation Query Command FCCOB Requirements**

| CCOBIX[2:0] | FCCOB Parameters | |
|---|---|---|
| 000 | 0x15 | Not required |
| 001 | Return DFPART | |
| 010 | Return ERPART | |
| 011 | Return ECOUNT(1) | |
| 100 | Return Dead Sector Count | Return Ready Sector Count |

1. Indicates sector erase count

Upon clearing CCIF to launch the EEPROM Emulation Query command, the CCIF flag will set after the EEE partition and status variables are stored in the FCCOBIX register.If the Emulation Query command is executed prior to partitioning (Partition D-Flash Command Section 29.4.2.15), the following reset values are returned: DFPART = 0x_FFFF, ERPART = 0x_FFFF, ECOUNT = 0x_FFFF, Dead Sector Count = 0x_00, Ready Sector Count = 0x_00.

**Table 29-76. EEPROM Emulation Query Command Error Handling**

| Register | Error Bit | Error Condition |
|---|---|---|
| FSTAT | ACCERR | Set if CCOBIX[2:0] != 000 at command launch |
| | | Set if a Load Data Field command sequence is currently active |
| | | Set if command not available in current mode (see Table 29-30) |
| | FPVIOL | None |
| | MGSTAT1 | None |
| | MGSTAT0 | None |
| FERSTAT | EPVIOLIF | None |

### 29.4.2.22 Partition D-Flash Command

The Partition D-Flash command allows the user to allocate sectors within the D-Flash block for applications and a partition within the buffer RAM for EEPROM access. The D-Flash block consists of 128 sectors with 256 bytes per sector. The Erase All Blocks command must be run prior to launching the Partition D-Flash command.

**Table 29-77. Partition D-Flash Command FCCOB Requirements**

| CCOBIX[2:0] | FCCOB Parameters | |
|---|---|---|
| 000 | 0x20 | Not required |
| 001 | Number of 256 byte sectors for the D-Flash user partition (DFPART) | |
| 010 | Number of 256 byte sectors for buffer RAM EEE partition (ERPART) | |

Upon clearing CCIF to launch the Partition D-Flash command, the following actions are taken to define a partition within the D-Flash block for direct access (DFPART) and a partition within the buffer RAM for EEE use (ERPART):

- Validate the DFPART and ERPART values provided:
  - DFPART <= 128 (maximum number of 256 byte sectors in D-Flash block)
  - ERPART <= 16 (maximum number of 256 byte sectors in buffer RAM)
  - If ERPART > 0, 128 - DFPART >= 12 (minimum number of 256 byte sectors in the D-Flash block required to support EEE)
  - If ERPART > 0, ((128-DFPART)/ERPART) >= 8 (minimum ratio of D-Flash EEE space to buffer RAM EEE space to support EEE)
- Erase verify the D-Flash block and the EEE nonvolatile information register
- Program DFPART to the EEE nonvolatile information register at global address 0x12_0000 (see Table 29-7)

- Program a duplicate DFPART to the EEE nonvolatile information register at global address 0x12_0002 (see Table 29-7)
- Program ERPART to the EEE nonvolatile information register at global address 0x12_0004 (see Table 29-7)
- Program a duplicate ERPART to the EEE nonvolatile information register at global address 0x12_0006 (see Table 29-7)

The D-Flash user partition will start at global address 0x10_0000. The buffer RAM EEE partition will end at global address 0x13_FFFF. After the Partition D-Flash operation has completed, the CCIF flag will set.

Running the Partition D-Flash command a second time will result in the ACCERR bit within the FSTAT register being set. The data value written corresponds to the number of 256 byte sectors allocated for either direct D-Flash access (DFPART) or buffer RAM EEE access (ERPART).

**Table 29-78. Partition D-Flash Command Error Handling**

| Register | Error Bit | Error Condition |
|---|---|---|
| FSTAT | ACCERR | Set if CCOBIX[2:0] != 010 at command launch |
| | | Set if a Load Data Field command sequence is currently active |
| | | Set if command not available in current mode (see Table 29-30) |
| | | Set if partitions have already been defined |
| | | Set if an invalid DFPART or ERPART selection is supplied |
| | FPVIOL | None |
| | MGSTAT1 | Set if any errors have been encountered during the read |
| | MGSTAT0 | Set if any non-correctable errors have been encountered during the read |
| FERSTAT | EPVIOLIF | None |

### 29.4.3 Interrupts

The Flash module can generate an interrupt when a Flash command operation has completed or when a Flash command operation has detected an EEE error or an ECC fault.

**Table 29-79. Flash Interrupt Sources**

| Interrupt Source | Interrupt Flag | Local Enable | Global (CCR) Mask |
|---|---|---|---|
| Flash Command Complete | CCIF (FSTAT register) | CCIE (FCNFG register) | I Bit |
| Flash EEE Erase Error | ERSERIF (FERSTAT register) | ERSERIE (FERCNFG register) | I Bit |
| Flash EEE Program Error | PGMERIF (FERSTAT register) | PGMERIE (FERCNFG register) | I Bit |
| Flash EEE Protection Violation | EPVIOLIF (FERSTAT register) | EPVIOLIE (FERCNFG register) | I Bit |
| Flash EEE Error Type 1 Violation | ERSVIF1 (FERSTAT register) | ERSVIE1 (FERCNFG register) | I Bit |
| Flash EEE Error Type 0 Violation | ERSVIF0 (FERSTAT register) | ERSVIE0 (FERCNFG register) | I Bit |
| ECC Double Bit Fault on Flash Read | DFDIF (FERSTAT register) | DFDIE (FERCNFG register) | I Bit |
| ECC Single Bit Fault on Flash Read | SFDIF (FERSTAT register) | SFDIE (FERCNFG register) | I Bit |

**NOTE**

Vector addresses and their relative interrupt priority are determined at the MCU level.

#### 29.4.3.1 Description of Flash Interrupt Operation

The Flash module uses the CCIF flag in combination with the CCIE interrupt enable bit to generate the Flash command interrupt request. The Flash module uses the ERSEIF, PGMEIF, EPVIOLIF, ERSVIF1, ERSVIF0, DFDIF and SFDIF flags in combination with the ERSEIE, PGMEIE, EPVIOLIE, ERSVIE1, ERSVIE0, DFDIE and SFDIE interrupt enable bits to generate the Flash error interrupt request. For a detailed description of the register bits involved, refer to Section 29.3.2.5, “Flash Configuration Register (FCNFG)”, Section 29.3.2.6, “Flash Error Configuration Register (FERCNFG)”, Section 29.3.2.7, “Flash Status Register (FSTAT)”, and Section 29.3.2.8, “Flash Error Status Register (FERSTAT)”.

The logic used for generating the Flash module interrupts is shown in Figure 29-27.

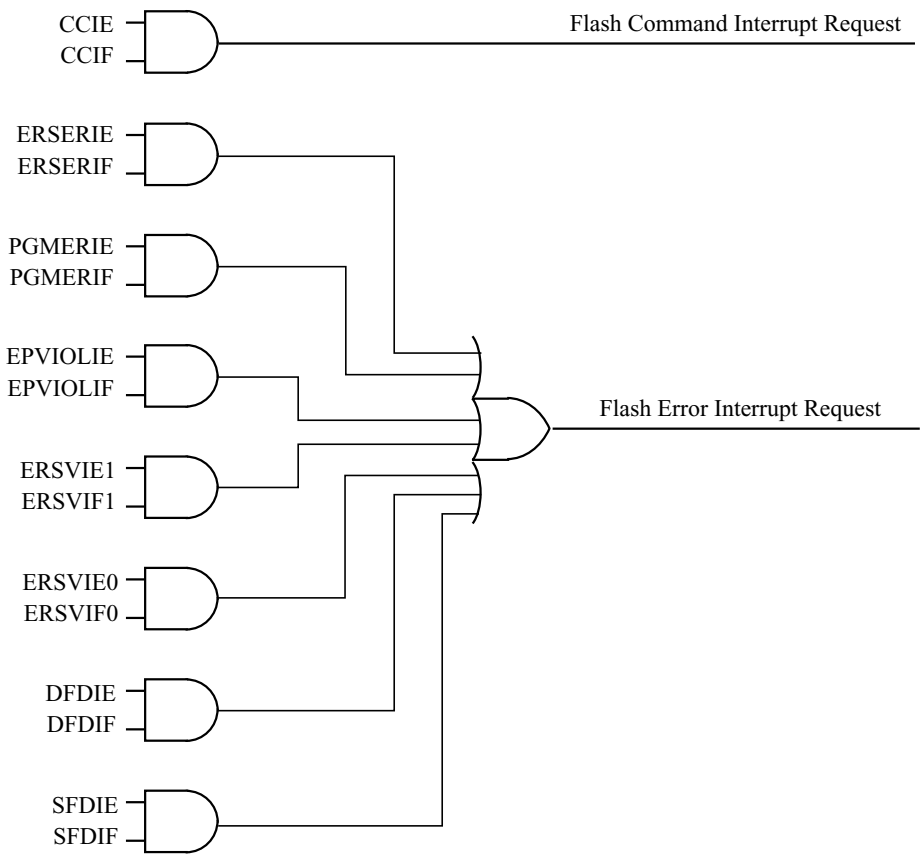



**Figure 29-27. Flash Module Interrupts Implementation**

### 29.4.4 Wait Mode

The Flash module is not affected if the MCU enters wait mode. The Flash module can recover the MCU from wait via the CCIF interrupt (see Section 29.4.3, “Interrupts”).

### 29.4.5 Stop Mode

If a Flash command is active (CCIF = 0) or an EE-Emulation operation is pending when the MCU requests stop mode, the current Flash operation will be completed before the CPU is allowed to enter stop mode.

## 29.5 Security

The Flash module provides security information to the MCU. The Flash security state is defined by the SEC bits of the FSEC register (see Table 29-12). During reset, the Flash module initializes the FSEC register using data read from the security byte of the Flash configuration field at global address 0x7F_FF0F.

The security state out of reset can be permanently changed by programming the security byte of the Flash configuration field. This assumes that you are starting from a mode where the necessary P-Flash erase and program commands are available and that the upper region of the P-Flash is unprotected. If the Flash security byte is successfully programmed, its new value will take affect after the next MCU reset.

The following subsections describe these security-related subjects:

- Unsecuring the MCU using Backdoor Key Access
- Unsecuring the MCU in Special Single Chip Mode using BDM
- Mode and Security Effects on Flash Command Availability

### 29.5.1 Unsecuring the MCU using Backdoor Key Access

The MCU may be unsecured by using the backdoor key access feature which requires knowledge of the contents of the backdoor keys (four 16-bit words programmed at addresses 0x7F_FF00–0x7F_FF07). If the KEYEN[1:0] bits are in the enabled state (see Section 29.3.2.2), the Verify Backdoor Access Key command (see Section 29.4.2.12) allows the user to present four prospective keys for comparison to the keys stored in the Flash memory via the Memory Controller. If the keys presented in the Verify Backdoor Access Key command match the backdoor keys stored in the Flash memory, the SEC bits in the FSEC register (see Table 29-12) will be changed to unsecure the MCU. Key values of 0x0000 and 0xFFFF are not permitted as backdoor keys. While the Verify Backdoor Access Key command is active, P-Flash block 0 will not be available for read access and will return invalid data.

The user code stored in the P-Flash memory must have a method of receiving the backdoor keys from an external stimulus. This external stimulus would typically be through one of the on-chip serial ports.

If the KEYEN[1:0] bits are in the enabled state (see Section 29.3.2.2), the MCU can be unsecured by the backdoor key access sequence described below:

1. Follow the command sequence for the Verify Backdoor Access Key command as explained in Section 29.4.2.12
2. If the Verify Backdoor Access Key command is successful, the MCU is unsecured and the SEC[1:0] bits in the FSEC register are forced to the unsecure state of 10

The Verify Backdoor Access Key command is monitored by the Memory Controller and an illegal key will prohibit future use of the Verify Backdoor Access Key command. A reset of the MCU is the only method to re-enable the Verify Backdoor Access Key command.

After the backdoor keys have been correctly matched, the MCU will be unsecured. After the MCU is unsecured, the sector containing the Flash security byte can be erased and the Flash security byte can be reprogrammed to the unsecure state, if desired.

In the unsecure state, the user has full control of the contents of the backdoor keys by programming addresses 0x7F_FF00–0x7F_FF07 in the Flash configuration field.

The security as defined in the Flash security byte (0x7F_FF0F) is not changed by using the Verify Backdoor Access Key command sequence. The backdoor keys stored in addresses 0x7F_FF00–0x7F_FF07 are unaffected by the Verify Backdoor Access Key command sequence. After the next reset of the MCU, the security state of the Flash module is determined by the Flash security byte

(0x7F_FF0F). The Verify Backdoor Access Key command sequence has no effect on the program and erase protections defined in the Flash protection register, FPROT.

### 29.5.2 Unsecuring the MCU in Special Single Chip Mode using BDM

The MCU can be unsecured in special single chip mode by erasing the P-Flash and D-Flash memory by one of the following methods:

- Reset the MCU into special single chip mode, delay while the erase test is performed by the BDM, send BDM commands to disable protection in the P-Flash and D-Flash memory, and execute the Erase All Blocks command write sequence to erase the P-Flash and D-Flash memory.
- Reset the MCU into special expanded wide mode, disable protection in the P-Flash and D-Flash memory and run code from external memory to execute the Erase All Blocks command write sequence to erase the P-Flash and D-Flash memory.

After the CCIF flag sets to indicate that the Erase All Blocks operation has completed, reset the MCU into special single chip mode. The BDM will execute the Erase Verify All Blocks command write sequence to verify that the P-Flash and D-Flash memory is erased. If the P-Flash and D-Flash memory are verified as erased the MCU will be unsecured. All BDM commands will be enabled and the Flash security byte may be programmed to the unsecure state by the following method:

- Send BDM commands to execute a ‘Program P-Flash’ command sequence to program the Flash security byte to the unsecured state and reset the MCU.

### 29.5.3 Mode and Security Effects on Flash Command Availability

The availability of Flash module commands depends on the MCU operating mode and security state as shown in Table 29-30.

## 29.6 Initialization

On each system reset the Flash module executes a reset sequence which establishes initial values for the Flash Block Configuration Parameters, the FPROT and DFPROT protection registers, and the FOPT and FSEC registers. The Flash module reverts to built-in default values that leave the module in a fully protected and secured state if errors are encountered during execution of the reset sequence. If a double bit fault is detected during the reset sequence, both MGSTAT bits in the FSTAT register will be set. The ACCERR bit in the FSTAT register is set if errors are encountered while initializing the EEE buffer ram during the reset sequence.

CCIF remains clear throughout the reset sequence. The Flash module holds off all CPU access for the initial portion of the reset sequence. While Flash reads are possible when the hold is removed, writes to the FCCOBIX, FCCOBHI, and FCCOBLO registers are ignored to prevent command activity while the Memory Controller remains busy. Completion of the reset sequence is marked by setting CCIF high which enables writes to the FCCOBIX, FCCOBHI, and FCCOBLO registers to launch any available Flash command.

If a reset occurs while any Flash command is in progress, that command will be immediately aborted. The state of the word being programmed or the sector/block being erased is not guaranteed.

# Appendix A
# Electrical Characteristics

## A.1 General

**NOTE**

The electrical characteristics given in this section should be used as a guide only. Values cannot be guaranteed by Freescale and are subject to change without notice.

This supplement contains the most accurate electrical information for the MC9S12XE-Family microcontroller available at the time of publication.

This introduction is intended to give an overview on several common topics like power supply, current injection etc.

### A.1.1 Parameter Classification

The electrical parameters shown in this supplement are guaranteed by various methods. To give the customer a better understanding the following classification is used and the parameters are tagged accordingly in the tables where appropriate.

**NOTE**

This classification is shown in the column labeled “C” in the parameter tables where appropriate.

P: Those parameters are guaranteed during production testing on each individual device.

C: Those parameters are achieved by the design characterization by measuring a statistically relevant sample size across process variations.

T: Those parameters are achieved by design characterization on a small sample size from typical devices under typical conditions unless otherwise noted. All values shown in the typical column are within this category.

D: Those parameters are derived mainly from simulations.

### A.1.2 Power Supply

The MC9S12XE-Family utilizes several pins to supply power to the I/O ports, A/D converter, oscillator, and PLL as well as the digital core.

The VDDA, VSSA pin pairs supply the A/D converter and parts of the internal voltage regulator.

The VDDX, VSSX pin pairs [7:1] supply the I/O pins.

VDDR supplies the internal voltage regulator.

**NOTE**

Connecting VDDR to VSS disables the internal voltage regulator.

The VDDF, VSS1 pin pair supplies the internal NVM logic.

The VDD, VSS2 are the supply pins for the internal digital logic.

VDDPLL, VSSPLL pin pair supply the oscillator and the PLL.

VSS1, VSS2 and VSS3 are internally connected by metal.

VDDA1, and VDDA2 are internally connected by metal.

All VDDX pins are internally connected by metal.

All VSSX pins are internally connected by metal.

VDDA is connected to all VDDX pins by diodes for ESD protection such that VDDX must not exceed VDDA by more than a diode voltage drop. VDDA can exceed VDDX by more than a diode drop in order to support applications with a 5V A/D converter range and 3.3V I/O pin range.
VSSA and VSSX are connected by anti-parallel diodes for ESD protection.

**NOTE**

In the following context $V_{DD35}$ is used for either VDDA, VDDR, and VDDX; $V_{SS35}$ is used for either VSSA and VSSX unless otherwise noted.

$I_{DD35}$ denotes the sum of the currents flowing into the VDDA and VDDR pins. The Run mode current in the VDDX domain is external load dependent.

$V_{DD}$ is used for VDD, $V_{SS}$ is used for VSS1, VSS2 and VSS3.

$V_{DDPLL}$ is used for VDDPLL, $V_{SSPLL}$ is used for VSSPLL

$I_{DD}$ is used for the sum of the currents flowing into VDD, VDDF and VDDPLL.

## A.1.3 Pins

There are four groups of functional pins.

### A.1.3.1 I/O Pins

Standard I/O pins have a level in the range of 3.13V to 5.5 V. This class of pins is comprised of all port I/O pins (including PortAD), BKGD and the $\overline{\text{RESET}}$ pins.The internal structure of all those pins is identical; however, some of the functionality may be disabled. For example the BKGD pin pull up is always enabled.

### A.1.3.2 Analog Reference

This group is made up by the $V_{RH}$ and $V_{RL}$ pins.

### A.1.3.3 Oscillator

The pins EXTAL, XTAL dedicated to the oscillator have a nominal 1.8 V level. They are supplied by VDDPLL.

### A.1.3.4 TEST

This pin is used for production testing only.

## A.1.4 Current Injection

Power supply must maintain regulation within operating $V_{DD35}$ or $V_{DD}$ range during instantaneous and operating maximum current conditions. If positive injection current ($V_{in} > V_{DD35}$) is greater than $I_{DD35}$, the injection current may flow out of $V_{DD35}$ and could result in external power supply going out of regulation. Ensure external $V_{DD35}$ load will shunt current greater than maximum injection current. This will be the greatest risk when the MCU is not consuming power; e.g., if no system clock is present, or if clock rate is very low which would reduce overall power consumption.

## A.1.5 Absolute Maximum Ratings

Absolute maximum ratings are stress ratings only. A functional operation under or outside those maxima is not guaranteed. Stress beyond those limits may affect the reliability or cause permanent damage of the device.

This device contains circuitry protecting against damage due to high static voltage or electrical fields; however, it is advised that normal precautions be taken to avoid application of any voltages higher than

maximum-rated voltages to this high-impedance circuit. Reliability of operation is enhanced if unused inputs are tied to an appropriate logic voltage level (e.g., either $V_{SS35}$ or $V_{DD35}$).

**Table A-1. Absolute Maximum Ratings(1)**

| Num | Rating | Symbol | Min | Max | Unit |
|---|---|---|---|---|---|
| 1 | I/O, regulator and analog supply voltage | $V_{DD35}$ | –0.3 | 6.0 | V |
| 2 | Digital logic supply voltage(2) | $V_{DD}$ | –0.3 | 2.16 | V |
| 3 | PLL supply voltage2 | $V_{DDPLL}$ | –0.3 | 2.16 | V |
| 4 | NVM supply voltage2 | $V_{DDF}$ | –0.3 | 3.6 | V |
| 5 | Voltage difference $V_{DDX}$ to $V_{DDA}$ | $\Delta_{VDDX}$ | –6.0 | 0.3 | V |
| 6 | Voltage difference $V_{SSX}$ to $V_{SSA}$ | $\Delta_{VSSX}$ | –0.3 | 0.3 | V |
| 7 | Digital I/O input voltage | $V_{IN}$ | –0.3 | 6.0 | V |
| 8 | Analog reference | $V_{RH}$, $V_{RL}$ | –0.3 | 6.0 | V |
| 9 | EXTAL, XTAL | $V_{ILV}$ | –0.3 | 2.16 | V |
| 10 | TEST input | $V_{TEST}$ | –0.3 | 10.0 | V |
| 11 | Instantaneous maximum current<br>Single pin limit for all digital I/O pins(3) | $I_D$ | –25 | +25 | mA |
| 12 | Instantaneous maximum current<br>Single pin limit for EXTAL, XTAL(4) | $I_{DL}$ | –25 | +25 | mA |
| 13 | Instantaneous maximum current<br>Single pin limit for TEST (5) | $I_{DT}$ | –0.25 | 0 | mA |
| 14 | Maximum current<br>Single pin limit for power supply pins | $I_{DV}$ | –100 | +100 | mA |
| 15 | Storage temperature range | $T_{stg}$ | –65 | 155 | °C |

1. Beyond absolute maximum ratings device might be damaged.
2. The device contains an internal voltage regulator to generate the logic and PLL supply out of the I/O supply. The absolute maximum ratings apply when the device is powered from an external source.
3. All digital I/O pins are internally clamped to $V_{SSX}$ and $V_{DDX}$, or $V_{SSA}$ and $V_{DDA}$.
4. Those pins are internally clamped to $V_{SSPLL}$ and $V_{DDPLL}$.
5. This pin is clamped low to $V_{SSPLL}$, but not clamped high. This pin must be tied low in applications.

## A.1.6 ESD Protection and Latch-up Immunity

All ESD testing is in conformity with CDF-AEC-Q100 stress test qualification for automotive grade integrated circuits. During the device qualification ESD stresses were performed for the Human Body Model (HBM) and the Charge Device Model.

A device will be defined as a failure if after exposure to ESD pulses the device no longer meets the device specification. Complete DC parametric and functional testing is performed per the applicable device

specification at room temperature followed by hot temperature, unless specified otherwise in the device specification.

**Table A-2. ESD and Latch-up Test Conditions**

| Model | Description | Symbol | Value | Unit |
|---|---|---|---|---|
| Human Body | Series resistance | R1 | 1500 | Ohm |
| | Storage capacitance | C | 100 | pF |
| | Number of pulse per pin<br>Positive<br>Negative | <br>—<br>— | <br>1<br>1 | |
| Charged Device | Number of pulse per pin<br>Positive<br>Negative | <br>—<br>— | <br>3<br>3 | |
| Latch-up | Minimum input voltage limit | | –2.5 | V |
| | Maximum input voltage limit | | 7.5 | V |

**Table A-3. ESD and Latch-Up Protection Characteristics**

| Num | C | Rating | Symbol | Min | Max | Unit |
|---|---|---|---|---|---|---|
| 1 | C | Human Body Model (HBM) | $V_{HBM}$ | 2000 | — | V |
| 2 | C | Charge Device Model (CDM) corner pins<br>Charge Device Model (CDM) edge pins | $V_{CDM}$ | 750<br>500 | —<br>— | V |
| 3 | C | Latch-up current at $T_A$ = 125°C<br>Positive<br>Negative | $I_{LAT}$ | <br>+100<br>–100 | <br>—<br>— | mA |
| 4 | C | Latch-up current at $T_A$ = 27°C<br>Positive<br>Negative | $I_{LAT}$ | <br>+200<br>–200 | <br>—<br>— | mA |

## A.1.7 Operating Conditions

This section describes the operating conditions of the device. Unless otherwise noted those conditions apply to all the following data.

**NOTE**

Please refer to the temperature rating of the device (C, V, M) with regards to the ambient temperature $T_A$ and the junction temperature $T_J$. For power dissipation calculations refer to Section A.1.8, "Power Dissipation and Thermal Characteristics".

**Table A-4. Operating Conditions**

| Rating | Symbol | Min | Typ | Max | Unit |
|---|---|---|---|---|---|
| I/O, regulator and analog supply voltage | $V_{DD35}$ | 3.13 | 5 | 5.5 | V |
| NVM logic supply voltage(1) | $V_{DDF}$ | 2.7 | 2.8 | 2.9 | V |
| Voltage difference $V_{DDX}$ to $V_{DDA}$ | $\Delta_{VDDX}$ | refer to Table A-15 | | | |
| Voltage difference $V_{DDR}$ to $V_{DDX}$ | $\Delta_{VDDR}$ | –0.1 | 0 | 0.1 | V |
| Voltage difference $V_{SSX}$ to $V_{SSA}$ | $\Delta_{VSSX}$ | refer to Table A-15 | | | |
| Voltage difference $V_{SS1}$ , $V_{SS2}$ , $V_{SS3}$ , $V_{SSPLL}$ to $V_{SSX}$ | $\Delta_{VSS}$ | –0.1 | 0 | 0.1 | V |
| Digital logic supply voltage[1] | $V_{DD}$ | 1.72 | 1.8 | 1.98 | V |
| PLL supply voltage | $V_{DDPLL}$ | 1.72 | 1.8 | 1.98 | V |
| Oscillator(2) (Loop Controlled Pierce)<br>(Full Swing Pierce) | $f_{osc}$ | 4<br>2 | —<br>— | 16<br>40 | MHz |
| Bus frequency(3) | $f_{bus}$ | 0.5 | — | 50 | MHz |
| **C** Operating junction temperature range<br>Operating ambient temperature range(4) | <br>$T_J$<br>$T_A$ | <br>–40<br>–40 | <br>—<br>27 | <br>110<br>85 | °C |
| **V** Operating junction temperature range<br>Operating ambient temperature range[2] | <br>$T_J$<br>$T_A$ | <br>–40<br>–40 | <br>—<br>27 | <br>130<br>105 | °C |
| **M** Operating junction temperature range<br>Operating ambient temperature range[2] | <br>$T_J$<br>$T_A$ | <br>–40<br>–40 | <br>—<br>27 | <br>150<br>125 | °C |

1. The device contains an internal voltage regulator to generate the logic and PLL supply out of the I/O supply. .
2. This refers to the oscillator base frequency. Typical crystal & resonator tolerances are supported.
3. Please refer to Table A-25 for maximum bus frequency limits with frequency modulation enabled
4. Please refer to Section A.1.8, "Power Dissipation and Thermal Characteristics" for more details about the relation between ambient temperature $T_A$ and device junction temperature $T_J$.

**NOTE**

Using the internal voltage regulator, operation is guaranteed in a power down until a low voltage reset assertion.

## A.1.8 Power Dissipation and Thermal Characteristics

Power dissipation and thermal characteristics are closely related. The user must assure that the maximum operating junction temperature is not exceeded. The average chip-junction temperature ($T_J$) in °C can be obtained from:

$$T_J = T_A + (P_D \bullet \Theta_{JA})$$

$T_J$ = Junction Temperature, [°C]

$T_A$ = Ambient Temperature, [°C]

$P_D$ = Total Chip Power Dissipation, [W]

$\Theta_{JA}$ = Package Thermal Resistance, [°C/W]

The total power dissipation can be calculated from:

$$P_D = P_{INT} + P_{IO}$$

$P_{INT}$ = Chip Internal Power Dissipation, [W]

$$P_{IO} = \sum_i R_{DSON} \cdot I_{IO_i}^2$$

$P_{IO}$ is the sum of all output currents on I/O ports associated with $V_{DDX}$, whereby

$$R_{DSON} = \frac{V_{OL}}{I_{OL}};\text{for outputs driven low}$$

$$R_{DSON} = \frac{V_{DD35} - V_{OH}}{I_{OH}};\text{for outputs driven high}$$

Two cases with internal voltage regulator enabled and disabled must be considered:

1. Internal voltage regulator disabled

$$P_{INT} = I_{DD} \cdot V_{DD} + I_{DDPLL} \cdot V_{DDPLL} + I_{DDA} \cdot V_{DDA}$$

2. Internal voltage regulator enabled

$$P_{INT} = I_{DDR} \cdot V_{DDR} + I_{DDA} \cdot V_{DDA}$$

**Table A-5. Thermal Package Characteristics (9S12XEP100)[1]**

| Num | C | Rating | Symbol | Min | Typ | Max | Unit |
|---|---|---|---|---|---|---|---|
| 208MAPBGA | | | | | | | |
| 1 | D | Thermal resistance 208MAPBGA, single sided PCB[2] | $\theta_{JA}$ | — | — | 53 | °C/W |
| 2 | D | Thermal resistance208MAPBGA, double sided PCB with 2 internal planes[3] | $\theta_{JA}$ | — | — | 31 | °C/W |
| 3 | D | Junction to Board 208MAPBGA[(2)] | $\theta_{JB}$ | — | — | 20 | °C/W |
| 4 | D | Junction to Case 208MAPBGA[4] | $\theta_{JC}$ | — | — | 9 | °C/W |
| 5 | D | Junction to Package Top 208MAPBGA[5] | $\Psi_{JT}$ | — | — | 2 | °C/W |
| LQFP144 | | | | | | | |
| 6 | D | Thermal resistance LQFP144, single sided PCB[3] | $\theta_{JA}$ | — | — | 41 | °C/W |
| 7 | D | Thermal resistance LQFP144, double sided PCB with 2 internal planes[3] | $\theta_{JA}$ | — | — | 32 | °C/W |
| 8 | D | Junction to Board LQFP 144 | $\theta_{JB}$ | — | — | 22 | °C/W |
| 9 | D | Junction to Case LQFP 144[4] | $\theta_{JC}$ | — | — | 7.4 | °C/W |
| 10 | D | Junction to Package Top LQFP144[5] | $\Psi_{JT}$ | — | — | 3 | °C/W |
| LQFP112 | | | | | | | |
| 11 | D | Thermal resistance LQFP112, single sided PCB[(3)] | $\theta_{JA}$ | — | — | 43 | °C/W |
| 12 | D | Thermal resistance LQFP112, double sided PCB with 2 internal planes[(4)] | $\theta_{JA}$ | — | — | 32 | °C/W |
| 13 | D | Junction to Board LQFP112 | $\theta_{JB}$ | — | — | 22 | °C/W |
| 14 | D | Junction to Case LQFP112[4] | $\theta_{JC}$ | — | — | 7 | °C/W |
| 15 | D | Junction to Package Top LQFP112[5] | $\Psi_{JT}$ | — | — | 3 | °C/W |
| QFP80 | | | | | | | |
| 16 | D | Thermal resistance QFP 80, single sided PCB[3] | $\theta_{JA}$ | — | — | 45 | °C/W |
| 17 | D | Thermal resistance QFP 80, double sided PCB with 2 internal planes[3] | $\theta_{JA}$ | — | — | 33 | °C/W |
| 18 | D | Junction to Board QFP 80 | $\theta_{JB}$ | — | — | 19 | °C/W |
| 19 | D | Junction to Case QFP 80[(5)] | $\theta_{JC}$ | — | — | 11 | °C/W |
| 20 | D | Junction to Package Top QFP 80[(6)] | $\Psi_{JT}$ | — | — | 3 | °C/W |

1. The values for thermal resistance are achieved by package simulations for the 9S12XEP100 die.
2. Measured per JEDEC JESD51-8. Measured on top surface of the board near the package.
3. Junction to ambient thermal resistance, $\theta_{JA}$ was simulated to be equivalent to the JEDEC specification JESD51-2 in a horizontal configuration in natural convection.
4. Junction to ambient thermal resistance, $\theta_{JA}$ was simulated to be equivalent to the JEDEC specification JESD51-7 in a horizontal configuration in natural convection.
5. Junction to case thermal resistance was simulated to be equivalent to the measured values using the cold plate technique with the cold plate temperature used as the “case” temperature. This basic cold plate measurement technique is described by MIL-STD 883D, Method 1012.1. This is the correct thermal metric to use to calculate thermal performance when the package is being used with a heat sink.
6. Thermal characterization parameter $\Psi_{JT}$ is the “resistance” from junction to reference point thermocouple on top center of the case as defined in JESD51-2. $\Psi_{JT}$ is a useful value to use to estimate junction temperature in a steady state customer enviroment.

**Table A-6. Thermal Package Characteristics (9S12XEQ512)(1)**

| Num | C | Rating | Symbol | Min | Typ | Max | Unit |
|---|---|---|---|---|---|---|---|
| LQFP144 | | | | | | | |
| 1a | D | Thermal resistance single sided PCB, natural convection | $\theta_{JA}$ | — | — | 49 | °C/W |
| 1b | D | Thermal resistance single sided PCB @ 200 ft/min | $\theta_{JA}$ | — | — | 40 | °C/W |
| 2a | D | Thermal resistance double sided PCB with 2 internal planes, natural convection | $\theta_{JA}$ | — | — | 40 | °C/W |
| 2b | D | Thermal resistance double sided PCB with 2 internal planes @ 200 ft/min | $\theta_{JA}$ | — | — | 34 | °C/W |
| 3 | D | Junction to Board LQFP 144 | $\theta_{JB}$ | — | — | 28 | °C/W |
| 4 | D | Junction to Case LQFP 144[2.] | $\theta_{JC}$ | — | — | 9 | °C/W |
| 5 | D | Junction to Package Top LQFP144[3] | $\Psi_{JT}$ | — | — | 2 | °C/W |
| LQFP112 | | | | | | | |
| 6a | D | Thermal resistance single sided PCB, natural convection | $\theta_{JA}$ | — | — | 50 | °C/W |
| 6b | D | Thermal resistance single sided PCB @ 200 ft/min | $\theta_{JA}$ | — | — | 40 | °C/W |
| 7a | D | Thermal resistance double sided PCB with 2 internal planes, natural convection | $\theta_{JA}$ | — | — | 40 | °C/W |
| 7b | D | Thermal resistance double sided PCB with 2 internal planes @ 200 ft/min | $\theta_{JA}$ | — | — | 34 | °C/W |
| 8 | D | Junction to Board LQFP112 | $\theta_{JB}$ | — | — | 28 | °C/W |
| 9 | D | Junction to Case LQFP112[2.] | $\theta_{JC}$ | — | — | 9 | °C/W |
| 10 | D | Junction to Package Top LQFP112[3] | $\Psi_{JT}$ | — | — | 2 | °C/W |
| QFP80 | | | | | | | |
| 11a | D | Thermal resistance single sided PCB, natural convection | $\theta_{JA}$ | — | — | 50 | °C/W |
| 11b | D | Thermal resistance single sided PCB @ 200 ft/min | $\theta_{JA}$ | — | — | 40 | °C/W |
| 12a | D | Thermal resistance double sided PCB with 2 internal planes, natural convection | $\theta_{JA}$ | — | — | 37 | °C/W |
| 12b | D | Thermal resistance double sided PCB with 2 internal planes @ 200 ft/min | $\theta_{JA}$ | — | — | 31 | °C/W |
| 13 | D | Junction to Board QFP 80 | $\theta_{JB}$ | — | — | 23 | °C/W |
| 14 | D | Junction to Case QFP 80(2) | $\theta_{JC}$ | — | — | 13 | °C/W |
| 15 | D | Junction to Package Top QFP 80(3) | $\Psi_{JT}$ | — | — | 3 | °C/W |

1. The values for thermal resistance are achieved by package simulations for the 9S12XEQ512 die.
2. Junction to case thermal resistance was simulated to be equivalent to the measured values using the cold plate technique with the cold plate temperature used as the “case” temperature. This basic cold plate measurement technique is described by MIL-STD 883D, Method 1012.1. This is the correct thermal metric to use to calculate thermal performance when the package is being used with a heat sink.
3. Thermal characterization parameter $\Psi_{JT}$ is the “resistance” from junction to reference point thermocouple on top center of the case as defined in JESD51-2. $\Psi_{JT}$ is a useful value to use to estimate junction temperature in a steady state customer enviroment.

## A.1.9 I/O Characteristics

This section describes the characteristics of all I/O pins except EXTAL, XTAL, TEST and supply pins.

**Table A-7. 3.3-V I/O Characteristics**

| Conditions are 3.13 V < $V_{DD35}$ < 3.6 V temperature from –40°C to +150°C, unless otherwise noted<br>I/O Characteristics for all I/O pins except EXTAL, XTAL,TEST and supply pins. | | | | | | | |
|---|---|---|---|---|---|---|---|
| **Num** | **C** | **Rating** | **Symbol** | **Min** | **Typ** | **Max** | **Unit** |
| 1 | P | Input high voltage | $V_{IH}$ | 0.65*$V_{DD35}$ | — | — | V |
| | T | Input high voltage | $V_{IH}$ | — | — | $V_{DD35}$ + 0.3 | V |
| 2 | P | Input low voltage | $V_{IL}$ | — | — | 0.35*$V_{DD35}$ | V |
| | T | Input low voltage | $V_{IL}$ | $V_{SS35}$ – 0.3 | — | — | V |
| 3 | T | Input hysteresis | $V_{HYS}$ | — | 250 | — | mV |
| 4a | P | Input leakage current (pins in high impedance input mode)[1] $V_{in}$ = $V_{DD35}$ or $V_{SS35}$<br>**M** Temperature range -40°C to 150°C<br>**V** Temperature range -40°C to 130°C<br>**C** Temperature range -40°C to 110°C | $I_{in}$ | <br>–1<br>–0.75<br>–0.5 | <br>—<br>—<br>— | <br>1<br>0.75<br>0.5 | µA |
| 4b | C | Input leakage current (pins in high impedance input mode) $V_{in}$ = $V_{DD35}$ or $V_{SS35}$<br>-40°C<br>27°C<br>70°C<br>85°C<br>100°C<br>105°C<br>110°C<br>120°C<br>125°C<br>130°C<br>150°C | $I_{in}$ | — | —<br>—<br>±1<br>±1<br>±8<br>±14<br>±26<br>±32<br>40<br>±60<br>±74<br>±92<br>±240 | — | nA |
| 5 | C | Output high voltage (pins in output mode)<br>Partial drive $I_{OH}$ = –0.75 mA | $V_{OH}$ | $V_{DD35}$ – 0.4 | — | — | V |
| 6 | P | Output high voltage (pins in output mode)<br>Full drive $I_{OH}$ = –4 mA | $V_{OH}$ | $V_{DD35}$ – 0.4 | — | — | V |
| 7 | C | Output low voltage (pins in output mode)<br>Partial Drive $I_{OL}$ = +0.9 mA | $V_{OL}$ | — | — | 0.4 | V |
| 8 | P | Output low voltage (pins in output mode)<br>Full Drive $I_{OL}$ = +4.75 mA | $V_{OL}$ | — | — | 0.4 | V |
| 9 | P | Internal pull up resistance<br>$V_{IH}$ min > input voltage > $V_{IL}$ max | $R_{PUL}$ | 25 | — | 50 | KΩ |
| 10 | P | Internal pull down resistance<br>$V_{IH}$ min > input voltage > $V_{IL}$ max | $R_{PDH}$ | 25 | — | 50 | KΩ |
| 11 | D | Input capacitance | $C_{in}$ | — | 6 | — | pF |
| 12 | T | Injection current[2]<br>Single pin limit<br>Total device limit, sum of all injected currents | <br>$I_{ICS}$<br>$I_{ICP}$ | <br>–2.5<br>–25 | — | <br>2.5<br>25 | mA |
| 13 | D | Port H, J, P interrupt input pulse filtered (STOP)[3] | $t_{PULSE}$ | — | — | 3 | µs |

**Table A-7. 3.3-V I/O Characteristics**

| Conditions are 3.13 V < $V_{DD35}$ < 3.6 V temperature from –40°C to +150°C, unless otherwise noted<br>I/O Characteristics for all I/O pins except EXTAL, XTAL,TEST and supply pins. | | | | | | | |
|---|---|---|---|---|---|---|---|
| 14 | D | Port H, J, P interrupt input pulse passed(STOP)[3] | $t_{PULSE}$ | 10 | — | — | µs |
| 15 | D | Port H, J, P interrupt input pulse filtered ($\overline{STOP}$) | $t_{PULSE}$ | — | — | 3 | tcyc |
| 16 | D | Port H, J, P interrupt input pulse passed($\overline{STOP}$) | $t_{PULSE}$ | 4 | — | — | tcyc |
| 17 | D | $\overline{IRQ}$ pulse width, edge-sensitive mode ($\overline{STOP}$) | $PW_{IRQ}$ | 1 | — | — | tcyc |
| 18 | D | $\overline{XIRQ}$ pulse width with X-bit set (STOP) | $PW_{XIRQ}$ | 4 | — | — | tosc |

1. Maximum leakage current occurs at maximum operating temperature.
2. Refer to Section A.1.4, “Current Injection” for more details
3. Parameter only applies in stop or pseudo stop mode.

## Table A-8. 5V I/O Characteristics

| Conditions are 4.5 V < $V_{DD35}$ < 5.5 V temperature from –40°C to +150°C, unless otherwise noted<br>I/O Characteristics for all I/O pins except EXTAL, XTAL,TEST and supply pins. | | | | | | | |
|---|---|---|---|---|---|---|---|
| Num | C | Rating | Symbol | Min | Typ | Max | Unit |
| 1 | P | Input high voltage | $V_{IH}$ | 0.65*$V_{DD35}$ | — | — | V |
| | T | Input high voltage | $V_{IH}$ | — | — | $V_{DD35}$ + 0.3 | V |
| 2 | P | Input low voltage | $V_{IL}$ | — | — | 0.35*$V_{DD35}$ | V |
| | T | Input low voltage | $V_{IL}$ | $V_{SS35}$ – 0.3 | — | — | V |
| 3 | T | Input hysteresis | $V_{HYS}$ | — | 250 | — | mV |
| 4a | P | Input leakage current (pins in high impedance input mode)[1] $V_{in}$ = $V_{DD35}$ or $V_{SS35}$<br>**M** Temperature range -40°C to 150°C<br>**V** Temperature range -40°C to 130°C<br>**C** Temperature range -40°C to 110°C | $I_{in}$ | <br><br>–1<br>–0.75<br>–0.5 | <br><br>—<br>—<br>— | <br><br>1<br>0.75<br>0.5 | µA |
| 4b | C | Input leakage current (pins in high impedance input mode) $V_{in}$ = $V_{DD35}$ or $V_{SS35}$<br>-40°C<br>27°C<br>70°C<br>85°C<br>100°C<br>105°C<br>110°C<br>120°C<br>125°C<br>130°C<br>150°C | $I_{in}$ | — | —<br>—<br>±1<br>±1<br>±8<br>±14<br>±26<br>±32<br>40<br>±60<br>±74<br>±92<br>±240 | — | nA |
| 5 | C | Output high voltage (pins in output mode)<br>Partial drive $I_{OH}$ = –2 mA | $V_{OH}$ | $V_{DD35}$ – 0.8 | — | — | V |
| 6 | P | Output high voltage (pins in output mode)<br>Full drive $I_{OH}$ = –10 mA | $V_{OH}$ | $V_{DD35}$ – 0.8 | — | — | V |
| 7 | C | Output low voltage (pins in output mode)<br>Partial drive $I_{OL}$ = +2 mA | $V_{OL}$ | — | — | 0.8 | V |
| 8 | P | Output low voltage (pins in output mode)<br>Full drive $I_{OL}$ = +10 mA | $V_{OL}$ | — | — | 0.8 | V |
| 9 | P | Internal pull up resistance<br>$V_{IH}$ min > input voltage > $V_{IL}$ max | $R_{PUL}$ | 25 | — | 50 | KΩ |
| 10 | P | Internal pull down resistance<br>$V_{IH}$ min > input voltage > $V_{IL}$ max | $R_{PDH}$ | 25 | — | 50 | KΩ |
| 11 | D | Input capacitance | $C_{in}$ | — | 6 | — | pF |
| 12 | T | Injection current[2]<br>Single pin limit<br>Total device Limit, sum of all injected currents | <br>$I_{ICS}$<br>$I_{ICP}$ | <br>–2.5<br>–25 | — | <br>2.5<br>25 | mA |
| 13 | P | Port H, J, P interrupt input pulse filtered(STOP)[3] | $t_{PULSE}$ | — | — | 3 | µs |
| 14 | P | Port H, J, P interrupt input pulse passed(STOP)[3] | $t_{PULSE}$ | 10 | — | — | µs |
| 15 | D | Port H, J, P interrupt input pulse filtered ($\overline{STOP}$) | $t_{PULSE}$ | — | — | 3 | tcyc |

**Table A-8. 5V I/O Characteristics**

| Conditions are 4.5 V < $V_{DD35}$ < 5.5 V temperature from –40°C to +150°C, unless otherwise noted<br>I/O Characteristics for all I/O pins except EXTAL, XTAL,TEST and supply pins. | | | | | | | |
|---|---|---|---|---|---|---|---|
| 16 | D | Port H, J, P interrupt input pulse passed ($\overline{STOP}$) | $t_{PULSE}$ | 4 | — | — | tcyc |
| 17 | D | $\overline{IRQ}$ pulse width, edge-sensitive mode ($\overline{STOP}$) | $PW_{IRQ}$ | 1 | — | — | tcyc |
| 18 | D | $\overline{XIRQ}$ pulse width with X-bit set (STOP) | $PW_{XIRQ}$ | 4 | — | — | tosc |

1. Maximum leakage current occurs at maximum operating temperature.
2. Refer to Section A.1.4, “Current Injection” for more details
3. Parameter only applies in stop or pseudo stop mode.

**Table A-9. Characteristics of Expantion Bus Inputs Port C, D, PE5, PE6, and PE7 for Reduced Input Voltage Thresholds**

| Conditions are 4.5 V < $V_{DD35}$ < 5.5 V Temperature from –40°C to +150°C, unless otherwise noted | | | | | | | |
|---|---|---|---|---|---|---|---|
| **Num** | **C** | **Rating** | **Symbol** | **Min** | **Typ** | **Max** | **Unit** |
| 1 | D | Input high voltage | $V_{IH}$ | 1.75 | — | — | V |
| 2 | D | Input low voltage | $V_{IL}$ | — | — | 0.75 | V |
| 3 | T | Input hysteresis | $V_{HYS}$ | — | 100 | — | mV |

## A.1.10 Supply Currents

This section describes the current consumption characteristics of the device family as well as the conditions for the measurements.

### A.1.10.1 Typical Run Current Measurement Conditions

Since the current consumption of the output drivers is load dependent, all measurements are without output loads and with minimum I/O activity. The currents are measured in single chip mode, S12XCPU code is executed from Flash and XGATE code is executed from RAM. $V_{DD35}$=5V, internal voltage regulator is enabled and the bus frequency is 50MHz using a 4-MHz oscillator in loop controlled Pierce mode.

Furthermore in expanded modes the currents flowing in the system are highly dependent on the load at the address, data, and control signals as well as on the duty cycle of those signals. No generally applicable numbers can be given. A very good estimate is to take the single chip currents and add the currents due to the external loads.

Since the DBG and BDM modules are typically not used in the end application, the supply current values for these modules is not specified.

An overhead of current consumption exisits independent of the listed modules, due to voltage regulation and clock logic that is not dedicated to a specific module. This is listed in the table row named “overhead”.

### A.1.10.2 Maximum Run Current Measurement Conditions

Currents are measured in single chip mode, S12XCPU and XGATE code is executed from RAM with $V_{DD35}$=5.5V, internal voltage regulator enabled and a 50MHz bus frequency from a 4-MHz input. Characterized parameters are derived using a 4MHz loop controlled Pierce oscillator. Production test parameters are tested with a 4MHz square wave oscillator.

### A.1.10.3 Current Conditions

Unbonded ports must be correctly initialized to prevent current consumption due to floating inputs. Typical Stop current is measured with $V_{DD35}$=5V, maximum Stop current is measured with $V_{DD35}$=5.5V. Pseudo Stop currents are measured with the oscillator configured for 4MHz LCP mode.

Table A-10. shows the configuration of the peripherals for typical run current.

**Table A-10. Module Configurations for Typical Run Supply Current $V_{DD35}$=5V**

| Peripheral | Configuration |
|---|---|
| S12XCPU | 420 cycle loop: 384 DBNE cycles plus subroutine entry to stimulate stacking (RAM access) |
| XGATE | XGATE fetches code from RAM, XGATE runs in an infinite loop, reading the Status and Flag registers of CAN's, SPI's, SCI's in sequence and doing some bit manipulation on the data |
| MSCAN | Configured to loop-back mode using a bit rate of 500kbit/s |
| SPI | Configured to master mode, continuously transmit data (0x55 or 0xAA) at 2Mbit/s |
| SCI | Configured into loop mode, continuously transmit data (0x55) at speed of 19200 baud |
| IIC | Operate in master mode and continuously transmit data (0x55 or 0xAA) at 100Kbit/s |
| PWM | Configured to toggle its pins at the rate of 1kHz |
| ECT | The peripheral shall be configured in output compare mode. Pulse accumulator and modulus counter enabled. |
| ATD | The peripheral is configured to operate at its maximum specified frequency and to continuously convert voltages on all input channels in sequence. |
| PIT | PIT is enabled, Micro-timer register 0 and 1 loaded with $0F and timer registers 0 to 3 are loaded with $03/07/0F/1F. |
| RTI | Enabled with RTI Control Register (RTICTL) set to $59 |
| Overhead | VREG supplying 1.8V from a 5V input voltage, core clock tree active, PLL on |

**Table A-11. Module Configurations for Maximum Run Supply Current $V_{DD35}$=5.5V**

| Peripheral | Configuration |
|---|---|
| S12XCPU | 420 cycle loop: 384 DBNE cycles plus subroutine entry to stimulate stacking (RAM access) |
| XGATE | XGATE fetches code from RAM, XGATE runs in an infinite loop, reading the Status and Flag registers of CAN's, SPI's, SCI's in sequence and doing some bit manipulation on the data |
| MSCAN | Configured to loop-back mode using a bit rate of 1Mbit/s |
| SPI | Configured to master mode, continuously transmit data (0x55 or 0xAA) at 4Mbit/s |
| SCI | Configured into loop mode, continuously transmit data (0x55) at speed of 57600 baud |
| IIC | Operate in master mode and continuously transmit data (0x55 or 0xAA) at 100Kbit/s |
| PWM | Configured to toggle its pins at the rate of 40kHz |
| ECT | The peripheral shall be configured in output compare mode. Pulse accumulator and modulus counter enabled. |
| ATD | The peripheral is configured to operate at its maximum specified frequency and to continuously convert voltages on all input channels in sequence. |
| Overhead | VREG supplying 1.8V from a 5V input voltage, PLL on |

**Table A-12. Module Run Supply Currents**

| Conditions are shown in Table A-10 at ambient temperature unless otherwise noted | | | | | | |
|---|---|---|---|---|---|---|
| **Num** | **C** | **Rating** | **Min** | **Typ** | **Max** | **Unit** |
| 1 | T | S12XCPU | — | 12.76 | — | mA |
| 2 | T | XGATE | — | 24.20 | — | |
| 3 | T | Each MSCAN | — | 1.05 | — | |
| 4 | T | Each SPI | — | 0.22 | — | |
| 5 | T | Each SCI | — | 0.28 | — | |
| 6 | T | Each IIC | — | 0.40 | — | |
| 7 | T | PWM | — | 0.55 | — | |
| 8 | T | ECT | — | 1.16 | — | |
| 9 | T | Each ATD | — | 0.82 | — | |
| 10 | T | PIT | — | 0.61 | — | |
| 11 | T | RTI | — | 0.17 | — | |
| 12 | T | Overhead | — | 35.56 | — | |

## Table A-13. Run and Wait Current Characteristics

| Conditions are shown in Table A-4 unless otherwise noted | | | | | | | |
|---|---|---|---|---|---|---|---|
| **Num** | **C** | **Rating** | **Symbol** | **Min** | **Typ** | **Max** | **Unit** |
| **Run supply current (No external load, Peripheral Configuration see Table A-11.)** | | | | | | | |
| 1 | P | Peripheral Set[1]<br>$f_{osc}$=4MHz, $f_{bus}$=50MHz | $I_{DD35}$ | — | — | 100 | mA |
| **Run supply current (No external load, Peripheral Configuration see Table A-10.)** | | | | | | | |
| 2 | | Peripheral Set[(1)] Devices S12XEP100, S12XEP768 | $I_{DD35}$ | | | | mA |
| | C | $f_{osc}$=4MHz, $f_{bus}$=50MHz | | — | 84 | — | |
| | T | $f_{osc}$=4MHz, $f_{bus}$=20MHz | | — | 43 | — | |
| | T | $f_{osc}$=4MHz, $f_{bus}$=8MHz | | — | 24 | — | |
| 2a | | Peripheral Set[1] All other devices | | | | | |
| | T | $f_{osc}$=4MHz, $f_{bus}$=50MHz | | — | 72 | — | mA |
| 3 | | Peripheral Set[(2)] | | | | | mA |
| | T | $f_{osc}$=4MHz, $f_{bus}$=50MHz | | — | 63 | — | |
| | T | $f_{osc}$=4MHz, $f_{bus}$=20MHz | | — | 35 | — | |
| | T | $f_{osc}$=4MHz, $f_{bus}$=8MHz | | — | 21 | — | |
| 4 | | Peripheral Set[(3)] | | | | | mA |
| | T | $f_{osc}$=4MHz, $f_{bus}$=50MHz | | — | 62 | — | |
| | T | $f_{osc}$=4MHz, $f_{bus}$=20MHz | | — | 34 | — | |
| | T | $f_{osc}$=4MHz, $f_{bus}$=8MHz | | — | 21 | — | |
| 5 | | Peripheral Set[(4)] | | | | | mA |
| | T | $f_{osc}$=4MHz, $f_{bus}$=50MHz | | — | 60 | — | |
| | T | $f_{osc}$=4MHz, $f_{bus}$=20MHz | | — | 33 | — | |
| | T | $f_{osc}$=4MHz, $f_{bus}$=8MHz | | — | 20 | — | |
| 6 | | Peripheral Set[(5)] | | | | | mA |
| | T | $f_{osc}$=4MHz, $f_{bus}$=50MHz | | — | 59 | — | |
| | T | $f_{osc}$=4MHz, $f_{bus}$=20MHz | | — | 33 | — | |
| | T | $f_{osc}$=4MHz, $f_{bus}$=8MHz | | — | 20 | — | |
| 7 | | Peripheral Set[(6)] | | | | | mA |
| | T | $f_{osc}$=4MHz, $f_{bus}$=50MHz | | — | 57 | — | |
| | T | $f_{osc}$=4MHz, $f_{bus}$=20MHz | | — | 33 | — | |
| | T | $f_{osc}$=4MHz, $f_{bus}$=8MHz | | — | 20 | — | |
| **Wait supply current** | | | | | | | |
| 8 | C | Peripheral Set[1] ,PLL on<br>XGATE executing code from RAM | $I_{DDW}$ | — | — | 85 | mA |
| 9 | | Peripheral Set[2] | | | | | |
| | T | $f_{osc}$=4MHz, $f_{bus}$=50MHz | | — | 50 | — | |
| | T | $f_{osc}$=4MHz, $f_{bus}$=8MHz | | — | 12 | — | |
| 10 | P | All modules disabled, RTI enabled, PLL off | | — | — | 10 | |

1. The following peripherals are on: ATD0/ATD1/ECT/IIC1/PWM/SPI0-SPI2/SCI0-SCI7/CAN0-CAN4/XGATE
2. The following peripherals are on: ATD0/ATD1/ECT/IIC1/PWM/SPI0-SPI2/SCI0-SCI7/CAN0-CAN4
3. The following peripherals are on: ATD0/ATD1/ECT/IIC1/PWM/SPI0-SPI2/SCI0-SCI7
4. The following peripherals are on: ATD0/ATD1/ECT/IIC1/PWM/SPI0-SPI2
5. The following peripherals are on: ATD0/ATD1/ECT/IIC1/PWM
6. The following peripherals are on: ATD0/ATD1/ECT/IIC1

**Table A-14. Pseudo Stop and Full Stop Current**

| Conditions are shown in Table A-4, junction temperature, unless otherwise noted | | | | | | | |
|---|---|---|---|---|---|---|---|
| **Num** | **C** | **Rating** | **Symbol** | **Min** | **Typ** | **Max** | **Unit** |
| **Pseudo stop current (API, RTI, and COP disabled) PLL off, LCP mode** | | | | | | | |
| 10 | C | –40°C | $I_{DDPS}$ | — | 175 | — | µA |
| | P | 27°C | | — | 185 | 255 | |
| | C | 70°C | | — | 255 | — | |
| | C | 85°C | | — | 305 | — | |
| | C | 105°C | | — | 455 | — | |
| | P | 110°C | | — | 505 | 2155 | |
| | P | 130°C | | — | 805 | 3655 | |
| | P | 150°C | | — | 1555 | 7655 | |
| **Pseudo stop current (API, RTI, and COP enabled) PLL off, LCP mode** | | | | | | | |
| 11 | C | 27°C | $I_{DDPS}$ | — | 205 | — | µA |
| | C | 70°C | | — | 275 | — | |
| | C | 85°C | | — | 325 | — | |
| | C | 105°C | | — | 475 | — | |
| | C | 125°C | | — | 810 | — | |
| | C | 150°C | | — | 1575 | — | |
| **Stop Current** | | | | | | | |
| 12 | C | –40°C | $I_{DDS}$ | — | 20 | — | µA |
| | P | 27°C | | — | 30 | 100 | |
| | C | 70°C | | — | 100 | — | |
| | C | 85°C | | — | 150 | — | |
| | C | 105°C | | — | 300 | — | |
| | P | 110°C | | — | 350 | 2000 | |
| | C | 125°C | | — | 550 | — | |
| | P | 130°C | | — | 650 | 3500 | |
| | P | 150°C | | — | 1400 | 7500 | |
| **Stop Current (API active)** | | | | | | | |
| 13 | T | –40°C | $I_{DDS}$ | — | 32 | — | µA |
| | T | 27°C | | — | 42 | — | |
| | T | 85°C | | — | 162 | — | |
| | T | 110°C | | — | 362 | — | |
| | T | 130°C | | — | 662 | — | |
| **Stop Current (one ATD active)** | | | | | | | |
| 14 | T | 27°C | $I_{DDS}$ | — | 300 | — | µA |
| | T | 85°C | | — | 420 | — | |
| | T | 125°C | | — | 820 | — | |

## A.2 ATD Characteristics

This section describes the characteristics of the analog-to-digital converter.

### A.2.1 ATD Operating Characteristics

The Table A-15 and Table A-16 show conditions under which the ATD operates.

The following constraints exist to obtain full-scale, full range results:

$$V_{SSA} \le V_{RL} \le V_{IN} \le V_{RH} \le V_{DDA}.$$

This constraint exists since the sample buffer amplifier can not drive beyond the power supply levels that it ties to. If the input level goes outside of this range it will effectively be clipped.

**Table A-15. ATD Operating Characteristics**

| Conditions are shown in Table A-4 unless otherwise noted, supply voltage 3.13V < $V_{DDA}$ < 5.5 V | | | | | | | |
|---|---|---|---|---|---|---|---|
| **Num** | **C** | **Rating** | **Symbol** | **Min** | **Typ** | **Max** | **Unit** |
| 1 | D | Reference potential<br>Low<br>High | <br>$V_{RL}$<br>$V_{RH}$ | <br>$V_{SSA}$<br>$V_{DDA}/2$ | <br>—<br>— | <br>$V_{DDA}/2$<br>$V_{DDA}$ | <br>V<br>V |
| 2 | D | Voltage difference $V_{DDX}$ to $V_{DDA}$ | $\Delta_{VDDX}$ | –2.35 | 0 | 0.1 | V |
| 3 | D | Voltage difference $V_{SSX}$ to $V_{SSA}$ | $\Delta_{VSSX}$ | –0.1 | 0 | 0.1 | V |
| 4 | C | Differential reference voltage | $V_{RH}$-$V_{RL}$ | 3.13 | 5.0 | 5.5 | V |
| 5 | C | ATD Clock Frequency (derived from bus clock via the prescaler) | $f_{ATDCLk}$ | 0.25 | — | 8.3 | MHz |
| 6 | P | ATD Clock Frequency in Stop mode (internal generated temperature and voltage dependent clock, ICLK) | | 0.6 | 1 | 1.7 | MHz |
| 7 | D | ADC conversion in stop, recovery time[1] | $t_{ATDSTPRCV}$ | — | — | 1.5 | us |
| 8 | D | ATD Conversion Period[2]<br>12 bit resolution:<br>10 bit resolution:<br>8 bit resolution: | <br>$N_{CONV12}$<br>$N_{CONV10}$<br>$N_{CONV8}$ | <br>20<br>19<br>17 | <br>—<br>—<br>— | <br>42<br>41<br>39 | <br>ATD clock Cycles |

1. When converting in Stop Mode (ICLKSTP=1) an ATD Stop Recovery time tATDSTPRCV is required to switch back to bus clock based ATDCLK when leaving Stop Mode. Do not access ATD registers during this time.
2. The minimum time assumes a sample time of 4 ATD clock cycles. The maximum time assumes a sample time of 24 ATD clock cycles and the discharge feature (SMP_DIS) enabled, which adds 2 ATD clock cycles.

### A.2.2 Factors Influencing Accuracy

Source resistance, source capacitance and current injection have an influence on the accuracy of the ATD. A further factor is that PortAD pins that are configured as output drivers switching.

#### A.2.2.1 Differential Reference Voltage

The accuracy is reduced if the differential reference voltage is less than 3.13V when using the ATD in the 3.3V range or if the differential reference voltage is less than 4.5V when using the ATD in the 5V range.

### A.2.2.2 Port AD Output Drivers Switching

PortAD output drivers switching can adversely affect the ATD accuracy whilst converting the analog voltage on other PortAD pins because the output drivers are supplied from the VDDA/VSSA ATD supply pins. Although internal design measures are implemented to minimize the affect of output driver noise, it is recommended to configure PortAD pins as outputs only for low frequency, low load outputs. The impact on ATD accuracy is load dependent and not specified. The values specified are valid under condition that no PortAD output drivers switch during conversion.

### A.2.2.3 Source Resistance

Due to the input pin leakage current as specified in Table A-8 in conjunction with the source resistance there will be a voltage drop from the signal source to the ATD input. The maximum source resistance $R_S$ specifies results in an error (10-bit resolution) of less than 1/2 LSB (2.5 mV) at the maximum leakage current. If device or operating conditions are less than worst case or leakage-induced error is acceptable, larger values of source resistance of up to 10Kohm are allowed.

### A.2.2.4 Source Capacitance

When sampling an additional internal capacitor is switched to the input. This can cause a voltage drop due to charge sharing with the external and the pin capacitance. For a maximum sampling error of the input voltage ≤ 1LSB (10-bit resolution), then the external filter capacitor, $C_f \geq 1024 * (C_{INS}–C_{INN})$.

### A.2.2.5 Current Injection

There are two cases to consider.

1. A current is injected into the channel being converted. The channel being stressed has conversion values of $3FF (in 10-bit mode) for analog inputs greater than $V_{RH}$ and $000 for values less than $V_{RL}$ unless the current is higher than specified as disruptive condition.
2. Current is injected into pins in the neighborhood of the channel being converted. A portion of this current is picked up by the channel (coupling ratio K), This additional current impacts the accuracy of the conversion depending on the source resistance.

The additional input voltage error on the converted channel can be calculated as:

$$V_{ERR} = K * R_S * I_{INJ}$$

with $I_{INJ}$ being the sum of the currents injected into the two pins adjacent to the converted channel.

**Table A-16. ATD Electrical Characteristics**

| Conditions are shown in Table A-4 unless otherwise noted | | | | | | | |
|---|---|---|---|---|---|---|---|
| **Num** | **C** | **Rating** | **Symbol** | **Min** | **Typ** | **Max** | **Unit** |
| 1 | C | Max input source resistance(1) | $R_S$ | — | — | 1 | KΩ |
| 2 | D | Total input capacitance Non sampling<br>Total input capacitance Sampling | $C_{INN}$<br>$C_{INS}$ | —<br>— | —<br>— | 10<br>16 | pF |
| 3 | D | Input internal Resistance | $R_{INA}$ | — | 5 | 15 | kΩ |
| 4 | C | Disruptive analog input current | $I_{NA}$ | –2.5 | — | 2.5 | mA |
| 5 | C | Coupling ratio positive current injection | $K_p$ | — | — | 1E-4 | A/A |
| 6 | C | Coupling ratio negative current injection | $K_n$ | — | — | 2E-3 | A/A |

1. Refer to A.2.2.3 for further information concerning source resistance

## A.2.3 ATD Accuracy

Table A-17 and Table A-18 specify the ATD conversion performance excluding any errors due to current injection, input capacitance and source resistance.

### A.2.3.1 ATD Accuracy Definitions

For the following definitions see also Figure A-1.
Differential non-linearity (DNL) is defined as the difference between two adjacent switching steps.

$$DNL(i) = \frac{V_i - V_{i-1}}{1LSB} - 1$$

The integral non-linearity (INL) is defined as the sum of all DNLs:

$$INL(n) = \sum_{i=1}^{n} DNL(i) = \frac{V_n - V_0}{1LSB} - n$$

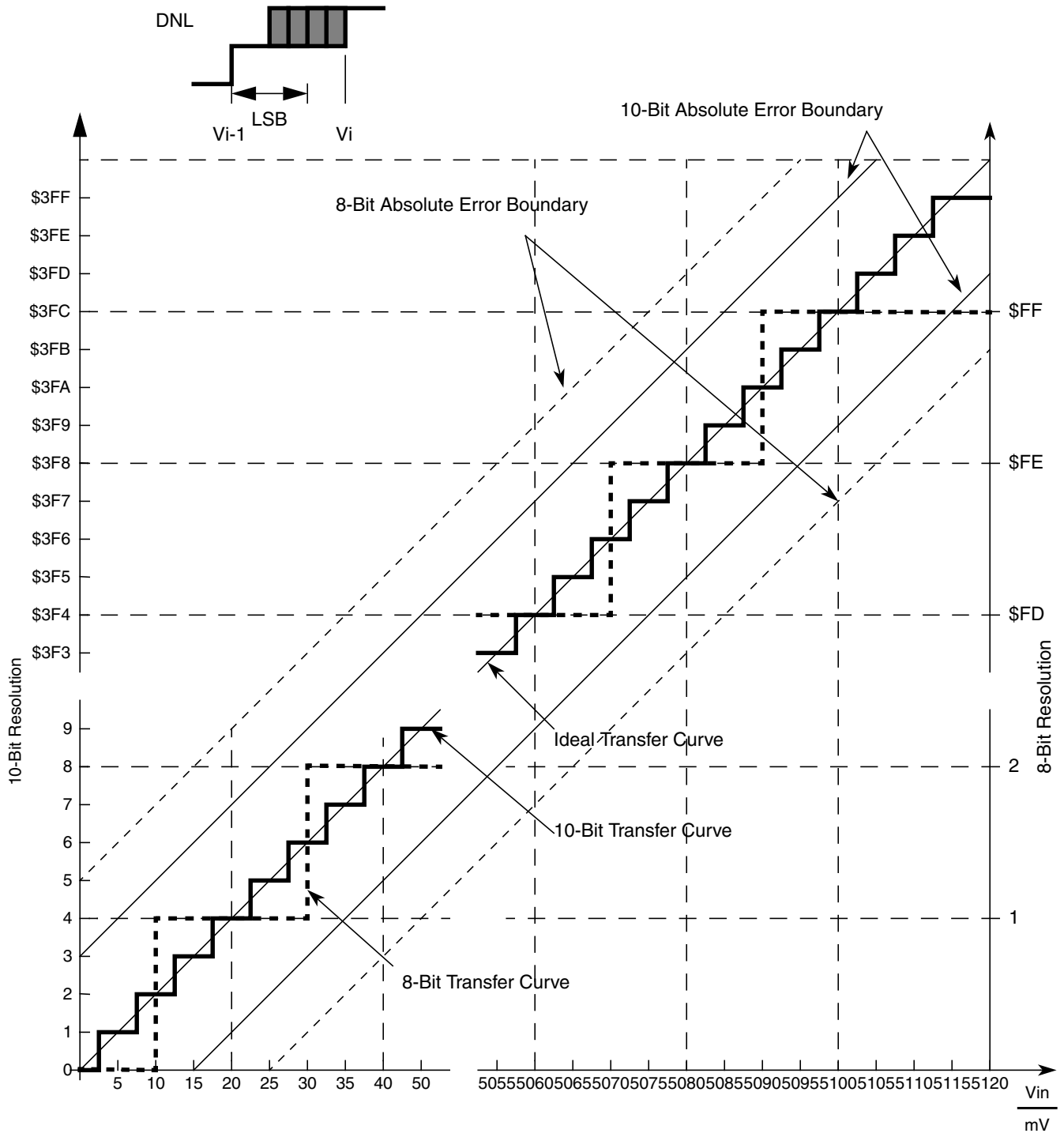



**Figure A-1. ATD Accuracy Definitions**

## NOTE

Figure A-1 shows only definitions, for specification values refer to Table A-17 and Table A-18

**Table A-17. ATD Conversion Performance 5V range**

| Conditions are shown in Table A-4. unless otherwise noted. $V_{REF} = V_{RH} - V_{RL} = 5.12V$. $f_{ATDCLK} = 8.3MHz$<br>The values are tested to be valid with no PortAD output drivers switching simultaneous with conversions. | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| **Num** | **C** | **Rating(1) ,(2)** | | **Symbol** | **Min** | **Typ** | **Max** | **Unit** |
| 1 | P | Resolution | 12-Bit | LSB | — | 1.25 | — | mV |
| 2 | P | Differential Nonlinearity | 12-Bit | DNL | -4 | ±2 | 4 | counts |
| 3 | P | Integral Nonlinearity | 12-Bit | INL | -5 | ±2.5 | 5 | counts |
| 4 | P | Absolute Error(3) | 12-Bit | AE | -7 | ±4 | 7 | counts |
| 5 | C | Resolution | 10-Bit | LSB | — | 5 | — | mV |
| 6 | C | Differential Nonlinearity | 10-Bit | DNL | -1 | ±0.5 | 1 | counts |
| 7 | C | Integral Nonlinearity | 10-Bit | INL | -2 | ±1 | 2 | counts |
| 8 | C | Absolute Error3. | 10-Bit | AE | -3 | ±2 | 3 | counts |
| 9 | C | Resolution | 8-Bit | LSB | — | 20 | — | mV |
| 10 | C | Differential Nonlinearity | 8-Bit | DNL | -0.5 | ±0.3 | 0.5 | counts |
| 11 | C | Integral Nonlinearity | 8-Bit | INL | -1 | ±0.5 | 1 | counts |
| 12 | C | Absolute Error3. | 8-Bit | AE | -1.5 | ±1 | 1.5 | counts |

1. The 8-bit and 10-bit mode operation is structurally tested in production test. Absolute values are tested in 12-bit mode.
2. Better performance is possible using specially designed multi-layer PCBs or averaging techniques.
3. These values include the quantization error which is inherently 1/2 count for any A/D converter.

**Table A-18. ATD Conversion Performance 3.3V range**

| Conditions are shown in Table A-4. unless otherwise noted. $V_{REF} = V_{RH} - V_{RL} = 3.3V$. $f_{ATDCLK} = 8.3MHz$<br>The values are tested to be valid with no PortAD output drivers switching simultaneous with conversions. | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| **Num** | **C** | **Rating(1),(2)** | | **Symbol** | **Min** | **Typ** | **Max** | **Unit** |
| 1 | P | Resolution | 12-Bit | LSB | — | 0.80 | — | mV |
| 2 | P | Differential Nonlinearity | 12-Bit | DNL | -6 | ±3 | 6 | counts |
| 3 | P | Integral Nonlinearity | 12-Bit | INL | -7 | ±3 | 7 | counts |
| 4 | P | Absolute Error(3) | 12-Bit | AE | -8 | ±4 | 8 | counts |
| 5 | C | Resolution | 10-Bit | LSB | — | 3.22 | — | mV |
| 6 | C | Differential Nonlinearity | 10-Bit | DNL | -1.5 | ±1 | 1.5 | counts |
| 7 | C | Integral Nonlinearity | 10-Bit | INL | -2 | ±1 | 2 | counts |
| 8 | C | Absolute Error3. | 10-Bit | AE | -3 | ±2 | 3 | counts |
| 9 | C | Resolution | 8-Bit | LSB | — | 12.89 | — | mV |
| 10 | C | Differential Nonlinearity | 8-Bit | DNL | -0.5 | ±0.3 | 0.5 | counts |
| 11 | C | Integral Nonlinearity | 8-Bit | INL | -1 | ±0.5 | 1 | counts |
| 12 | C | Absolute Error3. | 8-Bit | AE | -1.5 | ±1 | 1.5 | counts |

1. The 8-bit and 10-bit mode operation is structurally tested in production test. Absolute values are tested in 12-bit mode.
2. Better performance is possible using specially designed multi-layer PCBs or averaging techniques.
3. These values include the quantization error which is inherently 1/2 count for any A/D converter.

# A.3 NVM, Flash and Emulated EEPROM

## A.3.1 Timing Parameters

The time base for all NVM program or erase operations is derived from the oscillator. A minimum oscillator frequency $f_{NVMOSC}$ is required for performing program or erase operations. The NVM modules do not have any means to monitor the frequency and will not prevent program or erase operation at frequencies above or below the specified minimum. When attempting to program or erase the NVM modules at a lower frequency, a full program or erase transition is not assured.

The program and erase operations are timed using a clock derived from the oscillator using the FCLKDIV register. The frequency of this clock must be set within the limits specified as $f_{NVMOP}$.

The minimum program and erase times shown in Table A-19 are calculated for maximum $f_{NVMOP}$ and maximum $f_{NVMBUS}$ unless otherwise shown. The maximum times are calculated for minimum $f_{NVMOP}$

### A.3.1.1 Erase Verify All Blocks (Blank Check) (FCMD=0x01)

The time it takes to perform a blank check is dependant on the location of the first non-blank word starting at relative address zero. It takes one bus cycle per phrase to verify plus a setup of the command. Assuming that no non blank location is found, then the erase verify all blocks is given by.

$$t_{check} = 33500 \cdot \frac{1}{f_{NVMBUS}}$$

### A.3.1.2 Erase Verify Block (Blank Check) (FCMD=0x02)

The time it takes to perform a blank check is dependant on the location of the first non-blank word starting at relative address zero. It takes one bus cycle per phrase to verify plus a setup of the command. Assuming that no non blank location is found, then the erase verify time for a single 256K NVM array is given by

$$t_{check} = 33500 \cdot \frac{1}{f_{NVMBUS}}$$

For a 128K NVM or D-Flash array the erase verify time is given by

$$t_{check} = 17200 \cdot \frac{1}{f_{NVMBUS}}$$

### A.3.1.3 Erase Verify P-Flash Section (FCMD=0x03)

The maximum time depends on the number of phrases being verified ($N_{VP}$)

$$t_{check} = (752 + N_{VP}) \cdot \frac{1}{f_{NVMBUS}}$$

### A.3.1.4 Read Once (FCMD=0x04)

The maximum read once time is given by

$$t = (400) \cdot \frac{1}{f_{NVMBUS}}$$

### A.3.1.5 Load Data Field (FCMD=0x05)

The maximum load data field time is given by

$$t = (450) \cdot \frac{1}{f_{NVMBUS}}$$

### A.3.1.6 Program P-Flash (FCMD=0x06)

The programming time for a single phrase of four P-Flash words + associated eight ECC bits is dependant on the bus frequency as a well as on the frequency $f_{NVMOP}$ and can be calculated according to the following formulas, whereby $N_{DLOAD}$ is the number of extra blocks being programmed by the Load Data Field command (DLOAD), i.e. programming 2,3,4 blocks using DLOAD, $N_{DLOAD}$ =1,2,3 respectively.

The typical phrase programming time can be calculated using the following equation

$$t_{bwpgm} = (128 + (12 \cdot N_{DLOAD})) \cdot \frac{1}{f_{NVMOP}} + (1725 + (510 \cdot N_{DLOAD})) \cdot \frac{1}{f_{NVMBUS}}$$

The maximum phrase programming time can be calculated using the following equation

$$t_{bwpgm} = (130 + (14 \cdot N_{DLOAD})) \cdot \frac{1}{f_{NVMOP}} + (2125 + (510 \cdot N_{DLOAD})) \cdot \frac{1}{f_{NVMBUS}}$$

### A.3.1.7 P-Flash Program Once (FCMD=0x07)

The maximum P-Flash Program Once time is given by

$$t_{bwpgm} \approx 162 \cdot \frac{1}{f_{NVMOP}} + 2400 \cdot \frac{1}{f_{NVMBUS}}$$

### A.3.1.8 Erase All Blocks (FCMD=0x08)

For S12XEP100, S12XEP768, S12XEQ512 and S12XEQ384 erasing all blocks takes:

$$t_{mass} \approx 100100 \cdot \frac{1}{f_{NVMOP}} + 70000 \cdot \frac{1}{f_{NVMBUS}}$$

For S12XET256, S12XEA256 and S12XEG128 erasing all blocks takes:

$$t_{mass} \approx 100100 \cdot \frac{1}{f_{NVMOP}} + 35000 \cdot \frac{1}{f_{NVMBUS}}$$

### A.3.1.9 Erase P-Flash Block (FCMD=0x09)

Erasing a 256K NVM block takes

$$t_{mass} \approx 100100 \cdot \frac{1}{f_{NVMOP}} + 70000 \cdot \frac{1}{f_{NVMBUS}}$$

Erasing a 128K NVM block takes

$$t_{mass} \approx 100100 \cdot \frac{1}{f_{NVMOP}} + 35000 \cdot \frac{1}{f_{NVMBUS}}$$

### A.3.1.10 Erase P-Flash Sector (FCMD=0x0A)

The typical time to erase a1024-byte P-Flash sector can be calculated using

$$t_{era} = \left(20020 \cdot \frac{1}{f_{NVMOP}}\right) + \left(700 \cdot \frac{1}{f_{NVMBUS}}\right)$$

The maximum time to erase a1024-byte P-Flash sector can be calculated using

$$t_{era} = \left(20020 \cdot \frac{1}{f_{NVMOP}}\right) + \left(1100 \cdot \frac{1}{f_{NVMBUS}}\right)$$

### A.3.1.11 Unsecure Flash (FCMD=0x0B)

The maximum time for unsecuring the flash is given by

$$t_{uns} = \left(100100 \cdot \frac{1}{f_{NVMOP}} + 70000 \cdot \frac{1}{f_{NVMBUS}}\right)$$

### A.3.1.12 Verify Backdoor Access Key (FCMD=0x0C)

The maximum verify backdoor access key time is given by

$$t = 400 \cdot \frac{1}{f_{NVMBUS}}$$

### A.3.1.13 Set User Margin Level (FCMD=0x0D)

The maximum set user margin level time is given by

$$t = 350 \cdot \frac{1}{f_{NVMBUS}}$$

### A.3.1.14 Set Field Margin Level (FCMD=0x0E)

The maximum set field margin level time is given by

$$t = 350 \cdot \frac{1}{f_{NVMBUS}}$$

### A.3.1.15 Full Partition D-Flash (FCMD=0x0F)

The maximum time for partitioning the D-flash (ERPART=16, DFPART=0) is given by :

$$t_{part} \approx 21800 \cdot \frac{1}{f_{NVMOP}} + 400000 \cdot \frac{1}{f_{NVMBUS}} + t_{mass}$$

### A.3.1.16 Erase Verify D-Flash Section (FCMD=0x10)

Erase Verify D-Flash for a given number of words $N_W$ is given by .

$$t_{check} \approx (840 + N_W) \cdot \frac{1}{f_{NVMBUS}}$$

### A.3.1.17 D-Flash Programming (FCMD=0x11)

D-Flash programming time is dependent on the number of words being programmed and their location with respect to a row boundary, because programming across a row boundary requires extra steps. The D-Flash programming time is specified for different cases (1,2,3,4 words and 4 words across a row boundary) at a 50MHz bus frequency. The typical programming time can be calculated using the following equation, whereby $N_w$ denotes the number of words; BC=0 if no boundary is crossed and BC=1 if a boundary is crossed.

$$t_{dpgm} = \left((15 + (54 \cdot N_w) + (16 \cdot BC)) \cdot \frac{1}{f_{NVMOP}}\right) + \left((460 + (640 \cdot N_W) + (500 \cdot BC)) \cdot \frac{1}{f_{NVMBUS}}\right)$$

The maximum programming time can be calculated using the following equation

$$t_{dpgm} = \left((15 + (56 \cdot N_w) + (16 \cdot BC)) \cdot \frac{1}{f_{NVMOP}}\right) + \left((460 + (840 \cdot N_W) + (500 \cdot BC)) \cdot \frac{1}{f_{NVMBUS}}\right)$$

### A.3.1.18 Erase D-Flash Sector (FCMD=0x12)

Typical D-Flash sector erase times are those expected on a new device, where no margin verify fails occur. They can be calculated using the following equation.

$$t_{eradf} \approx 5025 \cdot \frac{1}{f_{NVMOP}} + 700 \cdot \frac{1}{f_{NVMBUS}}$$

Maximum D-Fash sector erase times can be calculated using the following equation.

$$t_{eradf} \approx 20100 \cdot \frac{1}{f_{NVMOP}} + 3300 \cdot \frac{1}{f_{NVMBUS}}$$

The D-Flash sector erase time on a new device is ~5ms and can extend to 20ms as the flash is cycled.

### A.3.1.19 Enable EEE (FCMD=0x13)

The maximum time to enable EPROM emulation is given by

$$t = \left(\left(((1100 \cdot BWN + (176 \cdot (1 + BWN) + (BWN + N_{SEC}) \cdot 32364))) \cdot \frac{1}{f_{NVMOP}}\right) + \left((3050 \cdot (1 + BWN) + (N_{SEC} + BWN) \cdot 290500) \cdot \frac{1}{f_{NVMBUS}}\right)\right)$$

where $N_{SEC}$ is the number of sectors of constant data. A constant sector is one in which all 63 records contain the latest active data and would need to be copied. The maximum possible is 33 (2048 EEE RAM words /63 =32.5) although this is a highly unlikely scenario. The impact of a worst case brownout recovery scenario is denoted by BWN = 2 for non brownout situations BWN =0.

### A.3.1.20 Maximum CCOB Latency

The maximum time a CCOB command has to wait to be actioned due to an EEE clean up is given where BWN = 1 if a brownout has occured otherwise BWN = 0. BWN = 1 only for the first ENEEE after reset.

$$t \approx \left(32364 \cdot \frac{1}{f_{NVMOP}} + 292600 \cdot \frac{1}{f_{NVMBUS}}\right) \cdot (1 + BWN) + BWN \cdot \left(350 \cdot \frac{1}{f_{NVMOP}} + \frac{1100}{f_{NVMBUS}}\right)$$

### A.3.1.21 Disable EEE (FCMD=0x14)

Maximum time to disable EPROM emulation is given by

$$t = 300 \cdot \frac{1}{f_{NVMBUS}}$$

### A.3.1.22 EEE Query (FCMD=0x15)

Maximum time for the EEE query command is given by

$$t = 300 \cdot \frac{1}{f_{NVMBUS}}$$

### A.3.1.23 Partition D-Flash (FCMD=0x20)

The maximum time for partitioning the D-flash (ERPART=16, DFPART=0) is given by

$$t \approx 21800 \cdot \frac{1}{f_{NVMOP}} + 400000 \cdot \frac{1}{f_{NVMBUS}}$$

### A.3.1.24 EEE Copy Down

The typical EEE copy down time is given by the following equation

$$t_{dfcd} = (14000 + (316 \cdot ERPART) + (1500 \cdot (124 - DFPART))) \times \frac{1}{f_{NVMBUS}}$$

The maximum EEE copy down time is given by the following equation

$$t_{dfcd} = (34000 + (316 \cdot ERPART) + (1500 \cdot (124 - DFPART))) \times \frac{1}{f_{NVMBUS}}$$

Worst case for Enable EEPROM Emulation allows for all the EEE records to have to be copied which is a very low probability scenario only likely in the case that the EEE is mostly full of unchanging data (the records for which are stored in consecutive D-Flash sectors).

**Table A-19. NVM Timing Characteristics**

| Conditions are as shown in Table A-4, with $f_{NVMBUS}$ = 50MHz and $f_{NVMOP}$= 1MHz unless otherwise noted. | | | | | | | |
|---|---|---|---|---|---|---|---|
| **Num** | **C** | **Rating** | **Symbol** | **Min** | **Typ** | **Max** | **Unit** |
| 1 | D | External oscillator clock | $f_{NVMOSC}$ | 2 | — | 50(1) | MHz |
| 2 | D | Bus frequency for programming or erase operations | $f_{NVMBUS}$ | 1 | — | 50 | MHz |
| 3 | D | Operating frequency | $f_{NVMOP}$ | 800 | — | 1050 | kHz |
| 4 | D | P-Flash phrase programming | $t_{bwpgm}$ | — | 162 | 173 | µs |
| 5a | D | P- Flash phrase program time using D-LOAD on 4 blocks | $t_{bwpgm4}$ | — | 231 | 264 | µs |
| 5b | D | P-Flash phrase program time using D-LOAD on 3 blocks | $t_{bwpgm3}$ | — | 208 | 233 | µs |
| 5c | D | P-Flash phrase program time using D-LOAD on 2 blocks | $t_{bwpgm2}$ | — | 185 | 202 | µs |
| 6 | P | P-Flash sector erase time | $t_{era}$ | — | 20 | 21 | ms |
| 7 | P | Erase All Blocks (Mass erase) time | $t_{mass}$ | — | 101 | 102 | ms |
| 7a | D | Unsecure Flash | $t_{uns}$ | — | 101 | 102 | ms |
| 8 | D | P-Flash erase verify (blank check) time(2) | $t_{check}$ | — | — | 33500[2] | $t_{cyc}$ |
| 9a | D | D-Flash word programming one word | $t_{dpgm}$ | — | 88 | 95 | µs |
| 9b | D | D-Flash word programming two words | $t_{dpgm}$ | — | 153 | 165 | µs |
| 9c | D | D-Flash word programming three words | $t_{dpgm}$ | — | 212 | 230 | µs |
| 9d | D | D-Flash word programming four words | $t_{dpgm}$ | — | 282 | 316 | µs |
| 9e | D | D-Flash word programming four words crossing row boundary | $t_{dpgm}$ | — | 298 | 342 | µs |
| 10 | D | D-Flash sector erase time | $t_{eradf}$ | — | 5.2(3) | 21 | ms |
| 11 | D | D-Flash erase verify (blank check) time | $t_{check}$ | — | — | 17500 | $t_{cyc}$ |
| 12 | D | EEE copy down (mask sets 5M48H, 3M25J, 2M53J, 1M12S, 1N35H, 1N36H) | $t_{dfrcd}$ | — | 255000 | 275000(4) | $t_{cyc}$ |
| 12 | D | EEE copy down (other mask sets) | $t_{dfrcd}$ | — | 205000 | 225000(5) | $t_{cyc}$ |

1. Restrictions for oscillator in crystal mode apply.
2. Valid for both “Erase verify all” or “Erase verify block” on 256K block without failing locations
3. This is a typical value for a new device
4. Maximum partitioning

5. Maximum partitioning

## A.3.2 NVM Reliability Parameters

The reliability of the NVM blocks is guaranteed by stress test during qualification, constant process monitors and burn-in to screen early life failures.

The data retention and program/erase cycling failure rates are specified at the operating conditions noted. The program/erase cycle count on the sector is incremented every time a sector or mass erase event is executed.

The standard shipping condition for both the D-Flash and P-Flash memory is erased with security disabled. However it is recommended that each block or sector is erased before factory programming to ensure that the full data retention capability is achieved. Data retention time is measured from the last erase operation.

**Table A-20. NVM Reliability Characteristics**

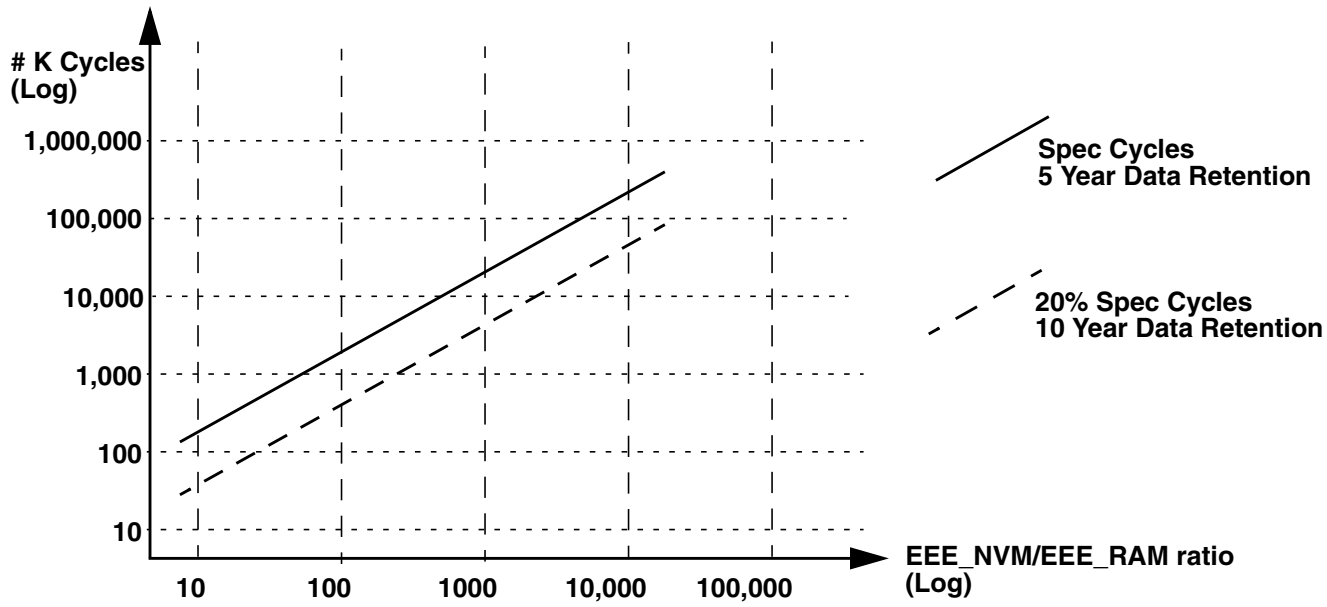
| Conditions are shown in Table A-4 unless otherwise noted | | | | | | | |
|---|---|---|---|---|---|---|---|
| **Num** | **C** | **Rating** | **Symbol** | **Min** | **Typ** | **Max** | **Unit** |
| | | **P-Flash Arrays** | | | | | |
| 1 | C | Data retention at an average junction temperature of $T_{Javg}$ = 85°C(1) after up to 10,000 program/erase cycles | $t_{PNVMRET}$ | 15 | 100(2) | — | Years |
| 2 | C | Data retention at an average junction temperature of $T_{Javg}$ = 85°C(3) after less than 100 program/erase cycles | $t_{PNVMRET}$ | 20 | 100[2] | — | Years |
| 3 | C | P-Flash number of program/erase cycles (-40°C ≤ tj ≤ 150°C) | $n_{PFLPE}$ | 10K | 100K[3] | — | Cycles |
| | | **D-Flash Array** | | | | | |
| 4 | C | Data retention at an average junction temperature of $T_{Javg}$ = 85°C[3] after up to 50,000 program/erase cycles | $t_{DNVMRET}$ | 5 | 100[2] | — | Years |
| 5 | C | Data retention at an average junction temperature of $T_{Javg}$ = 85°C[3] after less than 10,000 program/erase cycles | $t_{DNVMRET}$ | 10 | 100[2] | — | Years |
| 6 | C | Data retention at an average junction temperature of $T_{Javg}$ = 85°C[3] after less than 100 program/erase cycles | $t_{DNVMRET}$ | 20 | 100[2] | — | Years |
| 7 | C | D-Flash number of program/erase cycles (-40°C ≤ tj ≤ 150°C) | $n_{DFLPE}$ | 50K | 500K[3] | — | Cycles |
| | | **Emulated EEPROM** | | | | | |
| 8 | C | Data retention at an average junction temperature of $T_{Javg}$ = 85°C[1] after spec. program/erase cycles | $t_{EENVMRET}$ | 5[4] | 100[2] | — | Years |
| 9 | C | Data retention at an average junction temperature of $T_{Javg}$ = 85°C[3] after less than 20% spec.program/erase cycles. (e.g. after <20,000 cycles / Spec 100,000 cycles) | $t_{EENVMRET}$ | 10 | 100[2] | — | Years |
| 10 | C | Data retention at an average junction temperature of $T_{Javg}$ = 85°C[3] after less than 0.2% spec. program/erase cycles (e.g. after < 200 cycles / Spec 100,000 cycles) | $t_{EENVMRET}$ | 20 | 100[2] | — | Years |
| 11 | C | EEPROM number of program/erase cycles with a ratio of EEE_NVM to EEE_RAM = 8 (-40°C ≤ tj ≤ 150°C) | $n_{EEPE}$ | 100K(4) | 1M(5) | — | Cycles |
| 12 | C | EEPROM number of program/erase cycles with a ratio of EEE_NVM to EEE_RAM = 128 (-40°C ≤ tj ≤ 150°C) | $n_{EEPE}$ | 3M[4] | 30M[5] | — | Cycles |
| 13 | C | EEPROM number of program/erase cycles with a ratio of EEE_NVM to EEE_RAM = 16384(6) (-40°C ≤ tj ≤ 150°C) | $n_{EEPE}$ | 325M[4] | 3.2G[5] | — | Cycles |

1. $T_{Javg}$ does not exceed 85°C in a typical temperature profile over the lifetime of a consumer, industrial or automotive application.
2. Typical data retention values are based on intrinsic capability of the technology measured at high temperature and de-rated to 25°C using the Arrhenius equation. For additional information on how Freescale defines Typical Data Retention, please refer to Engineering Bulletin EB618
3. $T_{Javg}$ does not exceed 85°C in a typical temperature profile over the lifetime of a consumer, industrial or automotive application.
4. This represents the number of writes of updated data words to the EEE_RAM partition. Minimum specification (endurance and data retention) of the Emulated EEPROM array is based on the minimum specification of the D-Flash array per item 6.

5. This represents the number of writes of updated data words to the EEE_RAM partition. Typical endurance performance for the Emulated EEPROM array is based on typical endurance performance and the EEE algorithm implemented on this product family. Spec. table quotes typical endurance evaluated at 25°C for this product family.
6. This is equivalent to using a single byte or aligned word in the EEE_RAM with 32K D-Flash allocated for EEEPROM

The number of program/erase cycles for the EEPROM/D-Flash depends upon the partitioning of D-Flash used for EEPROM Emulation. Defining RAM size allocated for EEE as EEE-RAM and D-Flash partition allocated to EEE as EEE_NVM, the minimum number of program/erase cycles is specified depending upon the ratio of EEE_NVM/EEE_RAM. The minimum ratio EEE_NVM/EEE_RAM =8.

**Figure A-2. Program/Erase Dependency on D-Flash Partitioning**



# A.4 Voltage Regulator

Device functionality is guaranteed on power down to the LVR assert level.

**Table A-21. Voltage Regulator Electrical Characteristics**

| Conditions are shown in Table A-4 unless otherwise noted | | | | | | | |
|---|---|---|---|---|---|---|---|
| **Num** | **C** | **Characteristic** | **Symbol** | **Min** | **Typical** | **Max** | **Unit** |
| 1 | P | Input Voltages | $V_{VDDR,A}$ | 3.13 | — | 5.5 | V |
| 2 | P | Output Voltage Core<br>Full Performance Mode<br>Reduced Power Mode (MCU STOP mode)<br>Shutdown Mode | $V_{DD}$ | 1.72<br>—<br>— | 1.84<br>1.6<br>—(1) | 1.98<br>—<br>— | V<br>V<br>V |
| 3 | P | Output Voltage Flash<br>Full Performance Mode<br>Reduced Power Mode (MCU STOP mode)<br>Shutdown Mode | $V_{DDF}$ | 2.6<br>—<br>— | 2.82<br>2.2<br>—1 | 2.9<br>—<br>— | V<br>V<br>V |
| 4 | P | Output Voltage PLL<br>Full Performance Mode<br>Reduced Power Mode (MCU STOP mode)<br>Shutdown Mode | $V_{DDPLL}$ | 1.72<br>—<br>— | 1.84<br>1.6<br>—1 | 1.98<br>—<br>— | V<br>V<br>V |
| 5 | P | Low Voltage Interrupt Asser Level (2)<br>Low Voltage InterruptDeassert Level | $V_{LVIA}$<br>$V_{LVID}$ | 4.04<br>4.19 | 4.23<br>4.38 | 4.40<br>4.49 | V<br>V |
| 6a | P | VDDX Low Voltage Reset Deassert (3) | $V_{LVRXD}$ | — | — | 3.13 | V |
| 6b | D | VDDX Low Voltage Reset assert 3 | $V_{LVRXA}$ | — | 3.02 | — | V |
| 6c | P | VDDX Low Voltage Reset assert 3 | $V_{LVRXA}$ | 2.97 | — | — | V |
| 7 | C | Trimmed API internal clock(4) $\Delta f / f_{nominal}$ | $df_{API}$ | - 5% | — | + 5% | — |
| 8 | D | The first period after enabling the counter by APIFE might be reduced by API start up delay | $t_{sdel}$ | — | — | 100 | us |
| 9 | T | Temperature Sensor Slope | $dV_{TS}$ | 5.05 | 5.25 | 5.45 | mV/°C |
| 10 | T | High Temperature Interrupt Assert (VREGHTTR=$88)(5)<br>High Temperature Interrupt Deassert (VREGHTTR=$88) | $T_{HTIA}$<br>$T_{HTID}$ | 120<br>110 | 132<br>122 | 144<br>134 | °C |
| 11 | T | Bandgap Reference Voltage | $V_{BG}$ | 1.13 | 1.21 | 1.32 | V |

1. Voltage Regulator Disabled. High Impedance Output
2. Monitors VDDA, active only in Full Performance Mode. Indicates I/O & ADC performance degradation due to low supply voltage.
3. Monitors VDDX, active only in Full Performance Mode. MCU is monitored by the POR in RPM (see Figure A-3)
4. The API Trimming bits must be set that the minimum period equals to 0.2 ms.
5. A hysteresis is guaranteed by design

**NOTE**

The LVR monitors the voltages VDDF and VDDX. If the voltage drops on these supplies to a level which could prohibit the correct function of the microcontroller, the LVR triggers.

# A.5 Output Loads

## A.5.1 Resistive Loads

The voltage regulator is intended to supply the internal logic and oscillator. It allows no external DC loads.

## A.5.2 Capacitive Loads

The capacitive loads are specified in Table A-22. Ceramic capacitors with X7R dielectricum are required.

**Table A-22. - Required Capacitive Loads**

| Num | Characteristic | Symbol | Min | Recommended | Max | Unit |
|---|---|---|---|---|---|---|
| 1 | VDD/VDDF external capacitive load | $C_{DDext}$ | 176 | 220 | 264 | nF |
| 3 | VDDPLL external capacitive load | $C_{DDPLLext}$ | 80 | 220 | 264 | nF |

## A.5.3 Chip Power-up and Voltage Drops

LVI (low voltage interrupt), POR (power-on reset) and LVRs (low voltage reset) handle chip power-up or drops of the supply voltage. Their function is shown in Figure A-3 .

**Figure A-3. MC9S12XE-Family - Chip Power-up and Voltage Drops (not scaled)**

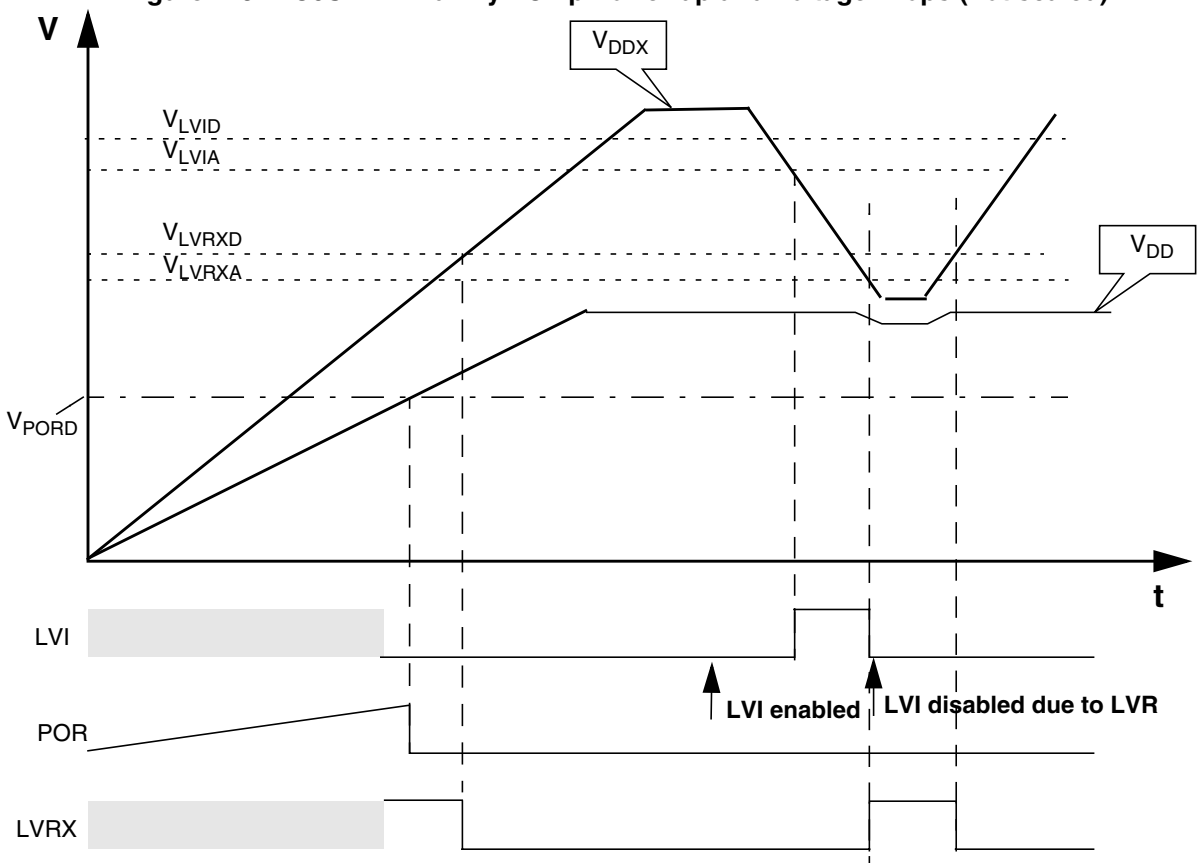
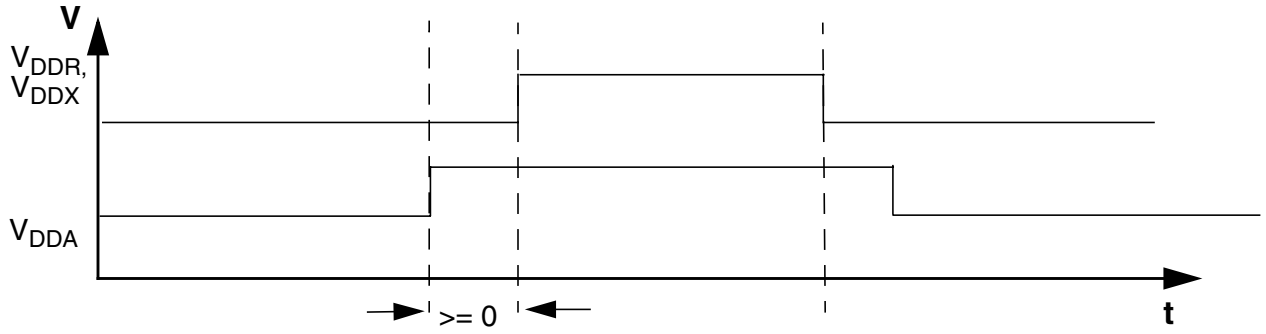


**Figure A-4. MC9S12XE-Family Power Sequencing**




During power sequencing $V_{DDA}$ can be powered up before $V_{DDR}$, $V_{DDX}$.
$V_{DDR}$ and $V_{DDX}$ must be powered up together adhering to the operating conditions differential.
$V_{RH}$ power up must follow $V_{DDA}$ to avoid current injection.

## A.6 Reset, Oscillator and PLL

This section summarizes the electrical characteristics of the various startup scenarios for oscillator and phase-locked loop (PLL).

### A.6.1 Startup

Table A-23 summarizes several startup characteristics explained in this section. Detailed description of the startup behavior can be found in the Clock and Reset Generator (CRG) block description

**Table A-23. Startup Characteristics**

| Conditions are shown in Table A-4unless otherwise noted | | | | | | | |
|---|---|---|---|---|---|---|---|
| **Num** | **C** | **Rating** | **Symbol** | **Min** | **Typ** | **Max** | **Unit** |
| 1 | D | Reset input pulse width, minimum input time | $PW_{RSTL}$ | 2 | — | — | $t_{osc}$ |
| 2 | D | Startup from reset | $t_{RST}$ | 192 | — | 4000(1) | $n_{bus}$ |
| 3 | D | Wait recovery startup time | $t_{WRS}$ | — | — | 14 | $t_{cyc}$ |
| 4 | D | Fast wakeup from STOP(2) | $t_{fws}$ | — | 50 | 100 | µs |

1. This is the time between RESET deassertion and start of CPU code execution.
2. Including voltage regulator startup; $V_{DD}$ /$V_{DDF}$ filter capacitors 220 nF, $V_{DD35}$ = 5 V, T= 25°C

#### A.6.1.1 POR

The release level $V_{PORR}$ and the assert level $V_{PORA}$ are derived from the $V_{DD}$ supply. They are also valid if the device is powered externally. After releasing the POR reset the oscillator and the clock quality check are started. If after a time $t_{CQOUT}$ no valid oscillation is detected, the MCU will start using the internal self clock. The fastest startup time possible is given by $n_{uposc}$.

### A.6.1.2 SRAM Data Retention

Provided an appropriate external reset signal is applied to the MCU, preventing the CPU from executing code when $V_{DD35}$ is out of specification limits, the SRAM contents integrity is guaranteed if after the reset the PORF bit in the CRG flags register has not been set.

### A.6.1.3 External Reset

When external reset is asserted for a time greater than $PW_{RSTL}$ the CRG module generates an internal reset, and the CPU starts fetching the reset vector without doing a clock quality check, if there was an oscillation before reset.

### A.6.1.4 Stop Recovery

Out of stop the controller can be woken up by an external interrupt. A clock quality check as after POR is performed before releasing the clocks to the system.

If the MCU is woken-up by an interrupt and the fast wake-up feature is enabled (FSTWKP = 1 and SCME = 1), the system will resume operation in self-clock mode after $t_{fws}$.

### A.6.1.5 Pseudo Stop and Wait Recovery

The recovery from pseudo stop and wait is essentially the same since the oscillator is not stopped in both modes. The controller can be woken up by internal or external interrupts. After $t_{wrs}$ the CPU starts fetching the interrupt vector.

## A.6.2 Oscillator

**Table A-24. Oscillator Characteristics**

| Conditions are shown in Table A-4. unless otherwise noted | | | | | | | |
|---|---|---|---|---|---|---|---|
| **Num** | **C** | **Rating** | **Symbol** | **Min** | **Typ** | **Max** | **Unit** |
| 1a | C | Crystal oscillator range (loop controlled Pierce) | $f_{OSC}$ | 4.0 | — | 16 | MHz |
| 1b | C | Crystal oscillator range (full swing Pierce) (1),(2) | $f_{OSC}$ | 2.0 | — | 40 | MHz |
| 2 | P | Startup Current | $i_{OSC}$ | 100 | — | — | µA |
| 3a | C | Oscillator start-up time (LCP, 4MHz)(3) | $t_{UPOSC}$ | — | 2 | 10 | ms |
| 3b | C | Oscillator start-up time (LCP, 8MHz)3 | $t_{UPOSC}$ | — | 1.6 | 8 | ms |
| 3c | C | Oscillator start-up time (LCP, 16MHz)3 | $t_{UPOSC}$ | — | 1 | 5 | ms |
| 4a | C | Oscillator start-up time (full swing Pierce, 2MHz)3 | $t_{UPOSC}$ | — | 8 | 40 | ms |
| 4b | C | Oscillator start-up time (full swing Pierce, 4MHz)3 | $t_{UPOSC}$ | — | 4 | 20 | ms |
| 4c | C | Oscillator start-up time (full swing Pierce, 8MHz)3 | $t_{UPOSC}$ | — | 2 | 10 | ms |
| 4d | C | Oscillator start-up time (full swing Pierce, 16MHz)3 | $t_{UPOSC}$ | — | 1 | 5 | ms |
| 4e | C | Oscillator start-up time (full swing Pierce, 40MHz)3 | $t_{UPOSC}$ | — | 0.8 | 4 | ms |
| 5 | D | Clock Quality check time-out | $t_{CQOUT}$ | 0.45 | — | 2.5 | s |
| 6 | P | Clock Monitor Failure Assert Frequency | $f_{CMFA}$ | 200 | 400 | 1000 | KHz |
| 7 | P | External square wave input frequency | $f_{EXT}$ | 2.0 | — | 50 | MHz |
| 8 | D | External square wave pulse width low | $t_{EXTL}$ | 9.5 | — | — | ns |
| 9 | D | External square wave pulse width high | $t_{EXTH}$ | 9.5 | — | — | ns |
| 10 | D | External square wave rise time | $t_{EXTR}$ | — | — | 1 | ns |
| 11 | D | External square wave fall time | $t_{EXTF}$ | — | — | 1 | ns |
| 12 | D | Input Capacitance (EXTAL, XTAL pins) | $C_{IN}$ | — | 7 | — | pF |
| 13 | P | EXTAL Pin Input High Voltage | $V_{IH,EXTAL}$ | $0.75*V_{DDPLL}$ | — | — | V |
| | T | EXTAL Pin Input High Voltage,(4) | $V_{IH,EXTAL}$ | — | — | $V_{DDPLL}$ + 0.3 | V |
| 14 | P | EXTAL Pin Input Low Voltage | $V_{IL,EXTAL}$ | — | — | $0.25*V_{DDPLL}$ | V |
| | T | EXTAL Pin Input Low Voltage,4 | $V_{IL,EXTAL}$ | $V_{SSPLL}$ - 0.3 | — | — | V |
| 15 | C | EXTAL Pin Input Hysteresis | $V_{HYS,EXTAL}$ | — | 180 | — | mV |
| 16 | C | EXTAL Pin oscillation amplitude (loop controlled Pierce) | $V_{PP,EXTAL}$ | — | 0.9 | — | V |

1. Depending on the crystal a damping series resistor might be necessary
2. Only valid if full swing Pierce oscillator/external clock mode is selected
3. These values apply for carefully designed PCB layouts with capacitors that match the crystal/resonator requirements..
4. Only applies if EXTAL is externally driven

## A.6.3 Phase Locked Loop

### A.6.3.1 Jitter Information

With each transition of the clock $f_{cmp}$, the deviation from the reference clock $f_{ref}$ is measured and input voltage to the VCO is adjusted accordingly.The adjustment is done continuously with no abrupt changes in the clock output frequency. Noise, voltage, temperature and other factors cause slight variations in the control loop resulting in a clock jitter. This jitter affects the real minimum and maximum clock periods as illustrated in Figure A-5.

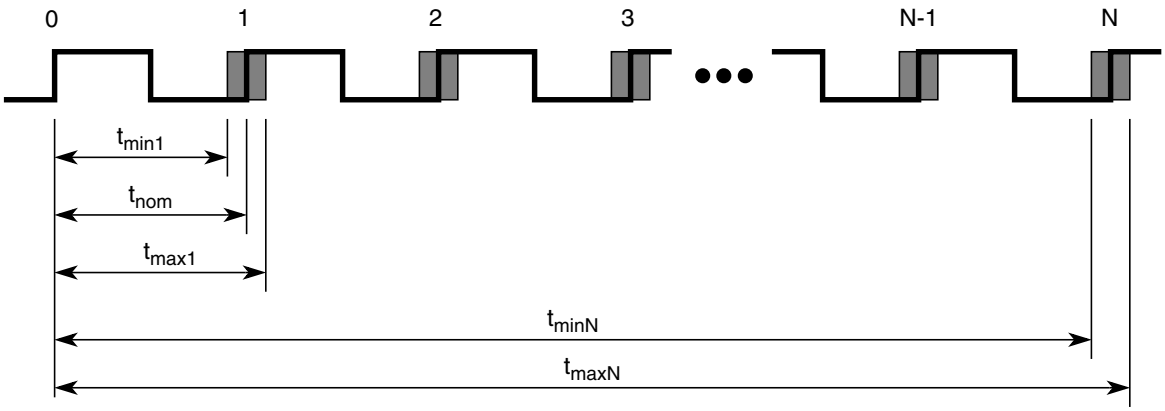



**Figure A-5. Jitter Definitions**

The relative deviation of $t_{nom}$ is at its maximum for one clock period, and decreases towards zero for larger number of clock periods (N).

Defining the jitter as:

$$J(N) = \max\left(\left|1 - \frac{t_{max}(N)}{N \cdot t_{nom}}\right|, \left|1 - \frac{t_{min}(N)}{N \cdot t_{nom}}\right|\right)$$

The following equation is a good fit for the maximum jitter:

$$J(N) = \frac{j_1}{\sqrt{N}} + j_2$$

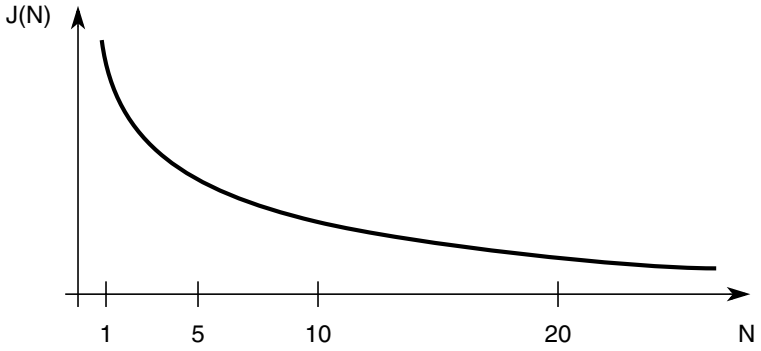



**Figure A-6. Maximum bus clock jitter approximation**

This is important to note with respect to timers, serial modules where a prescaler will eliminate the effect of the jitter to a large extent.

**Table A-25. IPLL Characteristics**

| Conditions are shown in Table A-4 unless otherwise noted | | | | | | | |
|---|---|---|---|---|---|---|---|
| **Num** | **C** | **Rating** | **Symbol** | **Min** | **Typ** | **Max** | **Unit** |
| 1 | P | Self Clock Mode frequency(1) | $f_{SCM}$ | 1 | — | 4 | MHz |
| 2 | C | VCO locking range | $f_{VCO}$ | 32 | — | 120 | MHz |
| 3 | C | Reference Clock | $f_{REF}$ | 1 | — | 40 | MHz |
| 4 | D | Lock Detection | $\|\Delta_{Lock}\|$ | 0 | — | 1.5 | %(2) |
| 5 | D | Un-Lock Detection | $\|\Delta_{unl}\|$ | 0.5 | — | 2.5 | %2 |
| 7 | C | Time to lock | $t_{lock}$ | — | 214 | 150 + 256/$f_{REF}$ | µs |
| 8 | C | Jitter fit parameter 1(3) | $j_1$ | — | — | 1.2 | % |
| 9 | C | Jitter fit parameter 23 | $j_2$ | — | — | 0 | % |
| 10 | D | Bus Frequency for FM1=1, FM0=1 (frequency modulation in PLLCTL register of s12xe_crg) | $f_{bus}$ | — | — | 48 | MHz |
| 11 | D | Bus Frequency for FM1=1, FM0=0 (frequency modulation in PLLCTL register of s12xe_crg) | $f_{bus}$ | — | — | 49 | MHz |
| 12 | D | Bus Frequency for FM1=0, FM0=1 (frequency modulation in PLLCTL register of s12xe_crg) | $f_{bus}$ | — | — | 49 | MHz |

1. Bus frequency is equivalent to $f_{SCM}/2$
2. % deviation from target frequency
3. $f_{OSC}$ = 4MHz, $f_{BUS}$ = 50MHz equivalent $f_{PLL}$ = 100MHz: REFDIV=$01, REFRQ=01, SYNDIV=$18, VCOFRQ=11, POSTDIV=$00.

# A.7 External Interface Timing

## A.7.1 MSCAN

**Table A-26. MSCAN Wake-up Pulse Characteristics**

| Conditions are shown in Table A-4 unless otherwise noted | | | | | | | |
|---|---|---|---|---|---|---|---|
| **Num** | **C** | **Rating** | **Symbol** | **Min** | **Typ** | **Max** | **Unit** |
| 1 | P | MSCAN wakeup dominant pulse filtered | $t_{WUP}$ | — | — | 1.5 | µs |
| 2 | P | MSCAN wakeup dominant pulse pass | $t_{WUP}$ | 5 | — | — | µs |

## A.7.2 SPI Timing

This section provides electrical parametrics and ratings for the SPI. In Table A-27 the measurement conditions are listed.

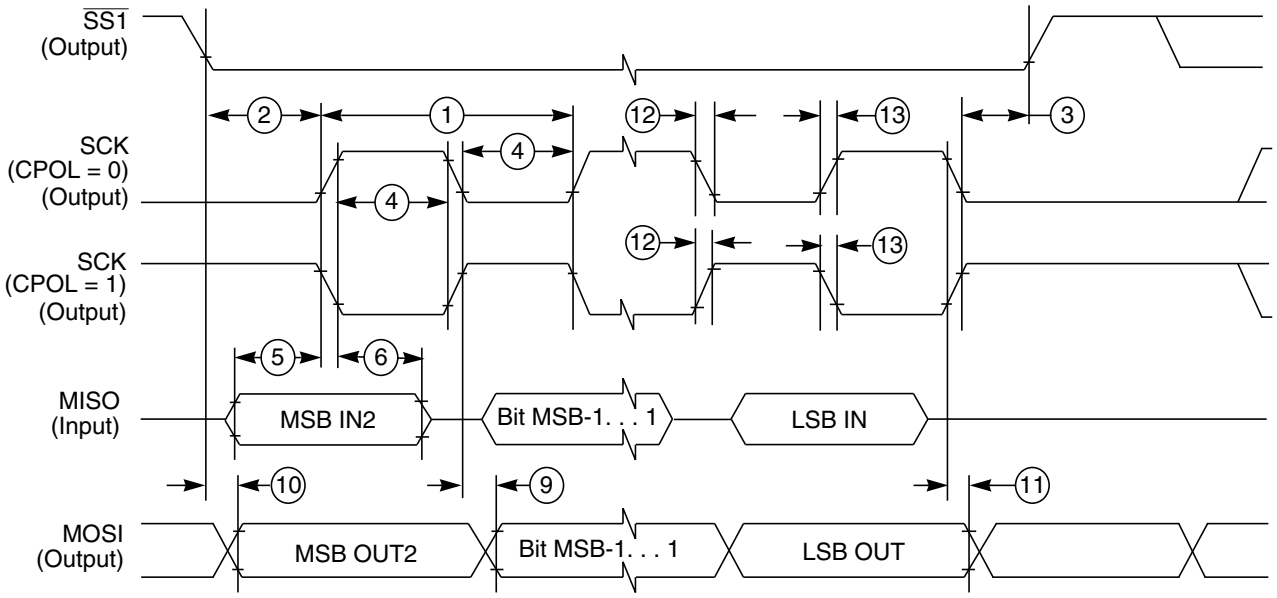
**Table A-27. Measurement Conditions**

| Description | Value | Unit |
|---|---|---|
| Drive mode | Full drive mode | — |
| Load capacitance $C_{LOAD}$[(1)], on all outputs | 50 | pF |
| Thresholds for delay measurement points | (20% / 80%) $V_{DDX}$ | V |

1. Timing specified for equal load on all SPI output pins. Avoid asymmetric load.

### A.7.2.1 Master Mode

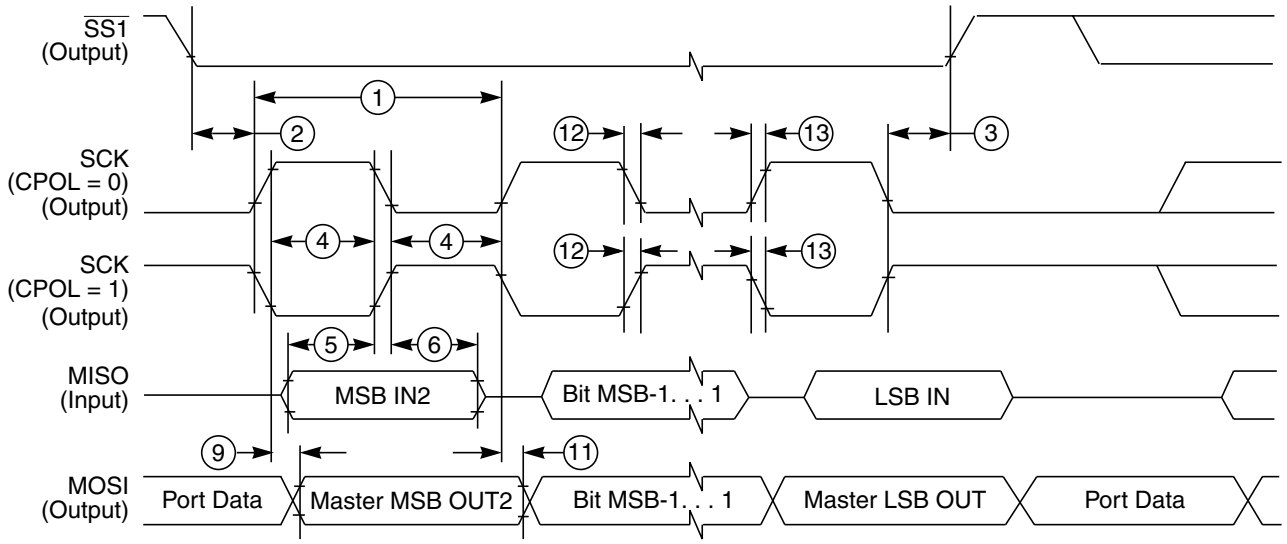
In Figure A-7 the timing diagram for master mode with transmission format CPHA = 0 is depicted.




1. If configured as an output.
2. LSBF = 0. For LSBF = 1, bit order is LSB, bit 1, bit 2... MSB.

**Figure A-7. SPI Master Timing (CPHA = 0)**

In Figure A-8 the timing diagram for master mode with transmission format CPHA=1 is depicted.




1.If configured as output
2. LSBF = 0. For LSBF = 1, bit order is LSB, bit 1,bit 2... MSB.

**Figure A-8. SPI Master Timing (CPHA = 1)**

In Table A-28 the timing characteristics for master mode are listed.

**Table A-28. SPI Master Mode Timing Characteristics**

| Num | C | Characteristic | Symbol | Min | Typ | Max | Unit |
|---|---|---|---|---|---|---|---|
| 1 | D | SCK frequency | $f_{sck}$ | 1/2048 | — | 1/2[1] | $f_{bus}$ |
| 1 | D | SCK period | $t_{sck}$ | $2^1$ | — | 2048 | $t_{bus}$ |
| 2 | D | Enable lead time | $t_{lead}$ | — | 1/2 | — | $t_{sck}$ |
| 3 | D | Enable lag time | $t_{lag}$ | — | 1/2 | — | $t_{sck}$ |
| 4 | D | Clock (SCK) high or low time | $t_{wsck}$ | — | 1/2 | — | $t_{sck}$ |
| 5 | D | Data setup time (inputs) | $t_{su}$ | 8 | — | — | ns |
| 6 | D | Data hold time (inputs) | $t_{hi}$ | 8 | — | — | ns |
| 9 | D | Data valid after SCK edge | $t_{vsck}$ | — | — | 15 | ns |
| 10 | D | Data valid after $\overline{SS}$ fall (CPHA = 0) | $t_{vss}$ | — | — | 15 | ns |
| 11 | D | Data hold time (outputs) | $t_{ho}$ | 0 | — | — | ns |
| 12 | D | Rise and fall time inputs | $t_{rfi}$ | — | — | 8 | ns |
| 13 | D | Rise and fall time outputs | $t_{rfo}$ | — | — | 8 | ns |

1. See **Figure A-9.**

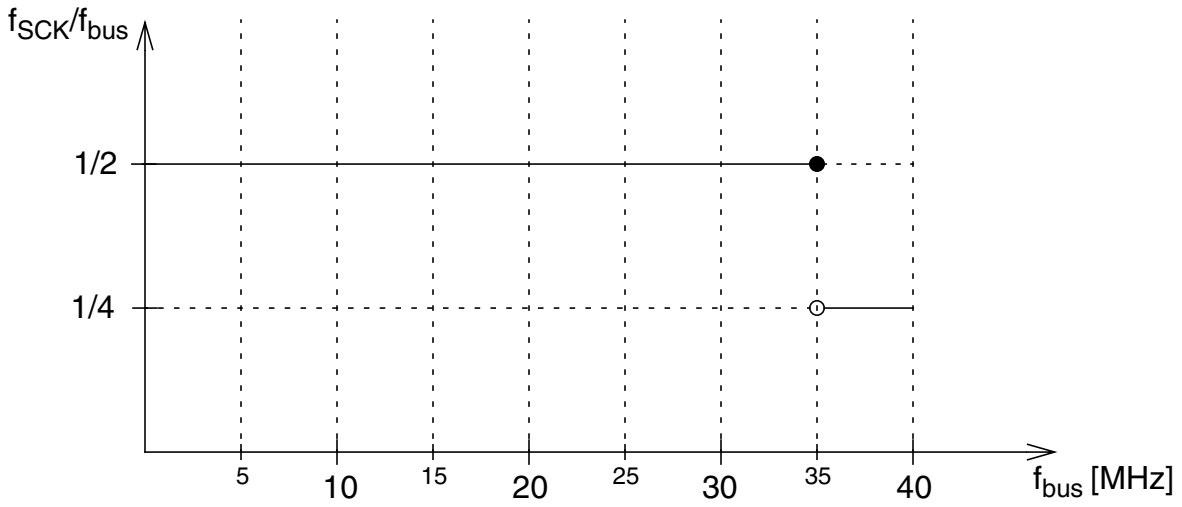
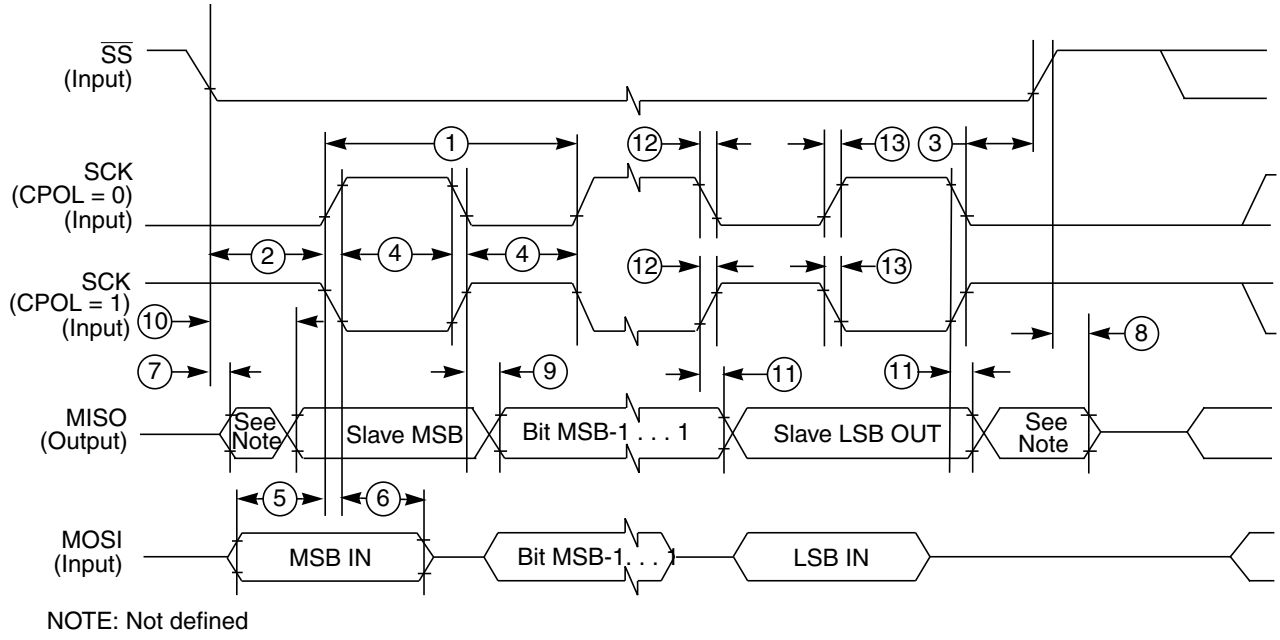



**Figure A-9. Derating of maximum $f_{SCK}$ to $f_{bus}$ ratio in Master Mode**

### A.7.2.2 Slave Mode

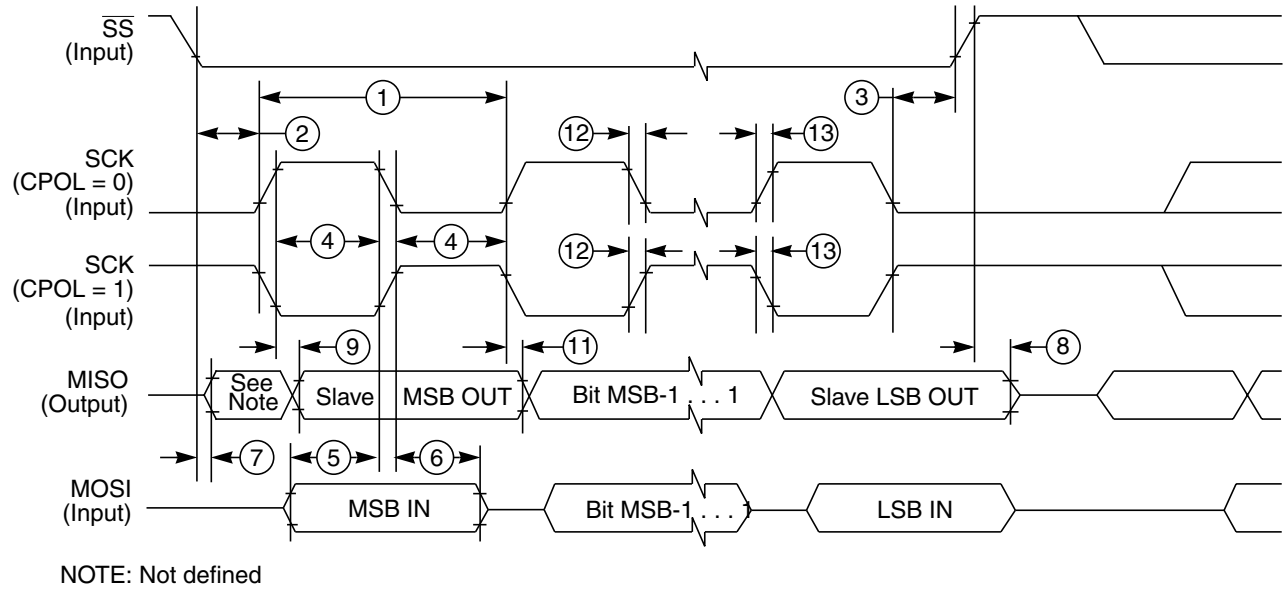
In Figure A-10 the timing diagram for slave mode with transmission format CPHA = 0 is depicted.




NOTE: Not defined

**Figure A-10. SPI Slave Timing (CPHA = 0)**

In Figure A-11 the timing diagram for slave mode with transmission format CPHA = 1 is depicted.




NOTE: Not defined

**Figure A-11. SPI Slave Timing (CPHA = 1)**

In Table A-29 the timing characteristics for slave mode are listed.

**Table A-29. SPI Slave Mode Timing Characteristics**

| Num | C | Characteristic | Symbol | Min | Typ | Max | Unit |
|---|---|---|---|---|---|---|---|
| 1 | D | SCK frequency | $f_{sck}$ | DC | — | 1/4 | $f_{bus}$ |
| 1 | D | SCK period | $t_{sck}$ | 4 | — | ∞ | $t_{bus}$ |
| 2 | D | Enable lead time | $t_{lead}$ | 4 | — | — | $t_{bus}$ |
| 3 | D | Enable lag time | $t_{lag}$ | 4 | — | — | $t_{bus}$ |
| 4 | D | Clock (SCK) high or low time | $t_{wsck}$ | 4 | — | — | $t_{bus}$ |
| 5 | D | Data setup time (inputs) | $t_{su}$ | 8 | — | — | ns |
| 6 | D | Data hold time (inputs) | $t_{hi}$ | 8 | — | — | ns |
| 7 | D | Slave access time (time to data active) | $t_{a}$ | — | — | 20 | ns |
| 8 | D | Slave MISO disable time | $t_{dis}$ | — | — | 22 | ns |
| 9 | D | Data valid after SCK edge | $t_{vsck}$ | — | — | $28 + 0.5 \cdot t_{bus}$ (1) | ns |
| 10 | D | Data valid after $\overline{SS}$ fall | $t_{vss}$ | — | — | $28 + 0.5 \cdot t_{bus}$ 1 | ns |
| 11 | D | Data hold time (outputs) | $t_{ho}$ | 20 | — | — | ns |
| 12 | D | Rise and fall time inputs | $t_{rfi}$ | — | — | 8 | ns |
| 13 | D | Rise and fall time outputs | $t_{rfo}$ | — | — | 8 | ns |

1. 0.5 $t_{bus}$ added due to internal synchronization delay

## A.7.3 External Bus Timing

The following conditions are assumed for all following external bus timing values:

- Crystal input within 45% to 55% duty
- Equal 25 pF load on all pins
- Pad full drive (reduced drive must be off)

### A.7.3.1 Normal Expanded Mode (External Wait Feature Disabled)

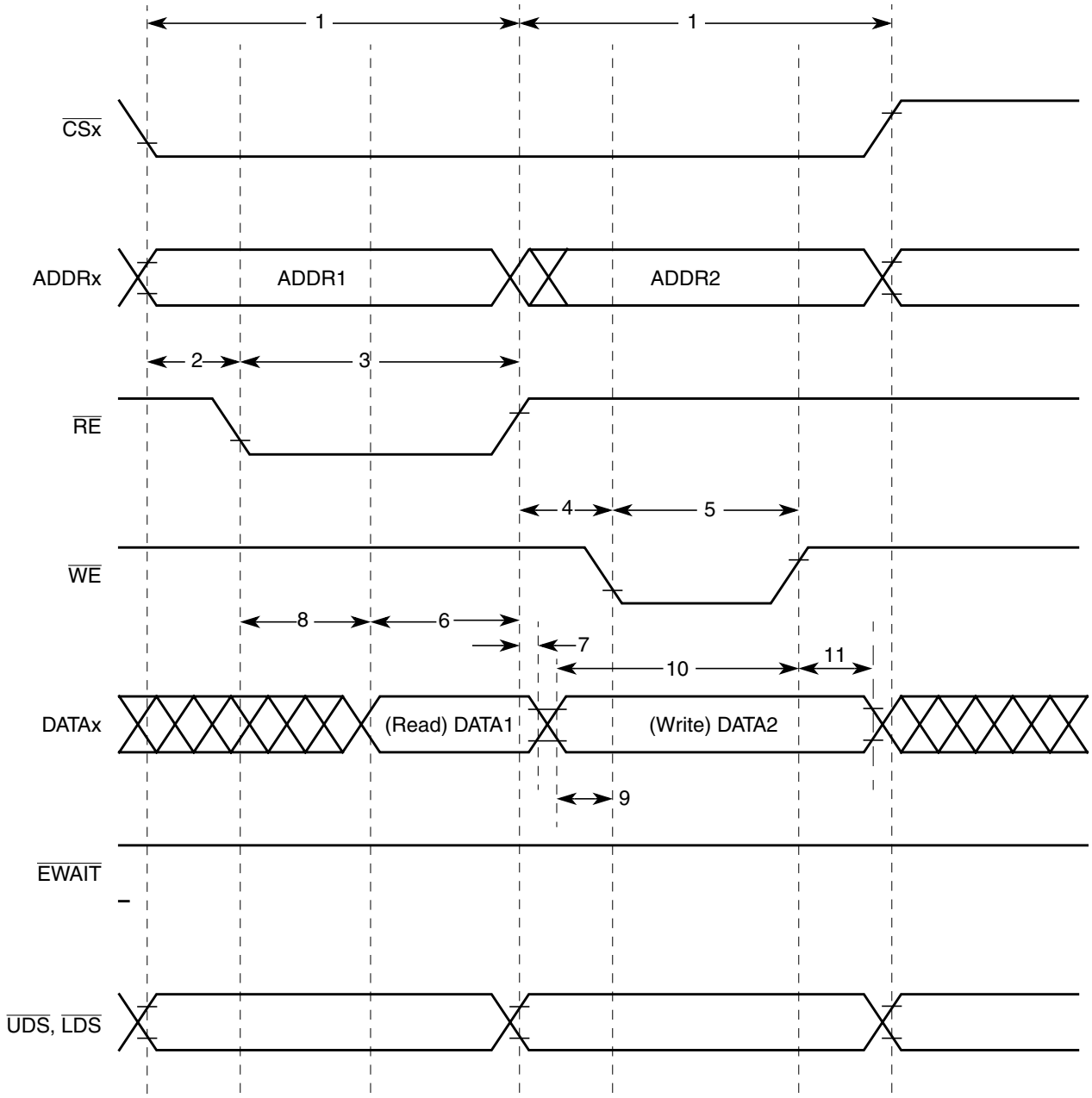



**Figure A-12. Example 1a: Normal Expanded Mode — Read Followed by Write**

**Table A-30. Example 1a: Normal Expanded Mode Timing 50 MHz bus (EWAIT disabled)**

| No. | Characteristic | Symbol | $V_{DD5}$=5.0V | | | $V_{DD5}$=3.3V | | | Unit |
|---|---|---|---|---|---|---|---|---|---|
| | | | C | Min | Max | C | Min | Max | |
| - | Frequency of internal bus | $f_i$ | - | D.C. | 50.0 | - | D.C. | 25.0 | MHz |
| - | Internal cycle time | $t_{cyc}$ | - | 20 | ∞ | - | 40 | ∞ | ns |
| - | Frequency of external bus | $f_o$ | - | D.C. | 25.0 | - | D.C. | 12.5 | MHz |
| 1 | External cycle time (selected by EXSTR) | $t_{cyce}$ | - | 40 | ∞ | - | 80 | ∞ | ns |
| 2 | Address [1] valid to $\overline{RE}$ fall | $t_{ADRE}$ | D | 4 | - | D | 13 | - | ns |
| 3 | Pulse width, $\overline{RE}$ | $PW_{RE}$ | D | 28 | - | D | 58 | - | ns |
| 4 | Address valid to $\overline{WE}$ fall | $t_{ADWE}$ | D | 4 | - | D | 15 | - | ns |
| 5 | Pulse width, $\overline{WE}$ | $PW_{WE}$ | D | 18 | - | D | 38 | - | ns |
| 6 | Read data setup time (if ITHRS = 0) | $t_{DSR}$ | D | 19 | - | D | 38 | - | ns |
| | Read data setup time (if ITHRS = 1) | $t_{DSR}$ | D | 23 | - | D | N/A | | ns |
| 7 | Read data hold time | $t_{DHR}$ | D | 0 | - | D | 0 | - | ns |
| 8 | Read enable access time | $t_{ACCR}$ | D | 4 | - | D | 4 | - | ns |
| 9 | Write data valid to $\overline{WE}$ fall | $t_{WDWE}$ | D | 5 | - | D | 5 | - | ns |
| 10 | Write data setup time | $t_{DSW}$ | D | 23 | - | D | 43 | - | ns |
| 11 | Write data hold time | $t_{DHW}$ | D | 6 | - | D | 4 | - | ns |

1. Includes the following signals: ADDRx, $\overline{UDS}$, $\overline{LDS}$, and $\overline{CSx}$.

### A.7.3.2 Normal Expanded Mode (External Wait Feature Enabled)

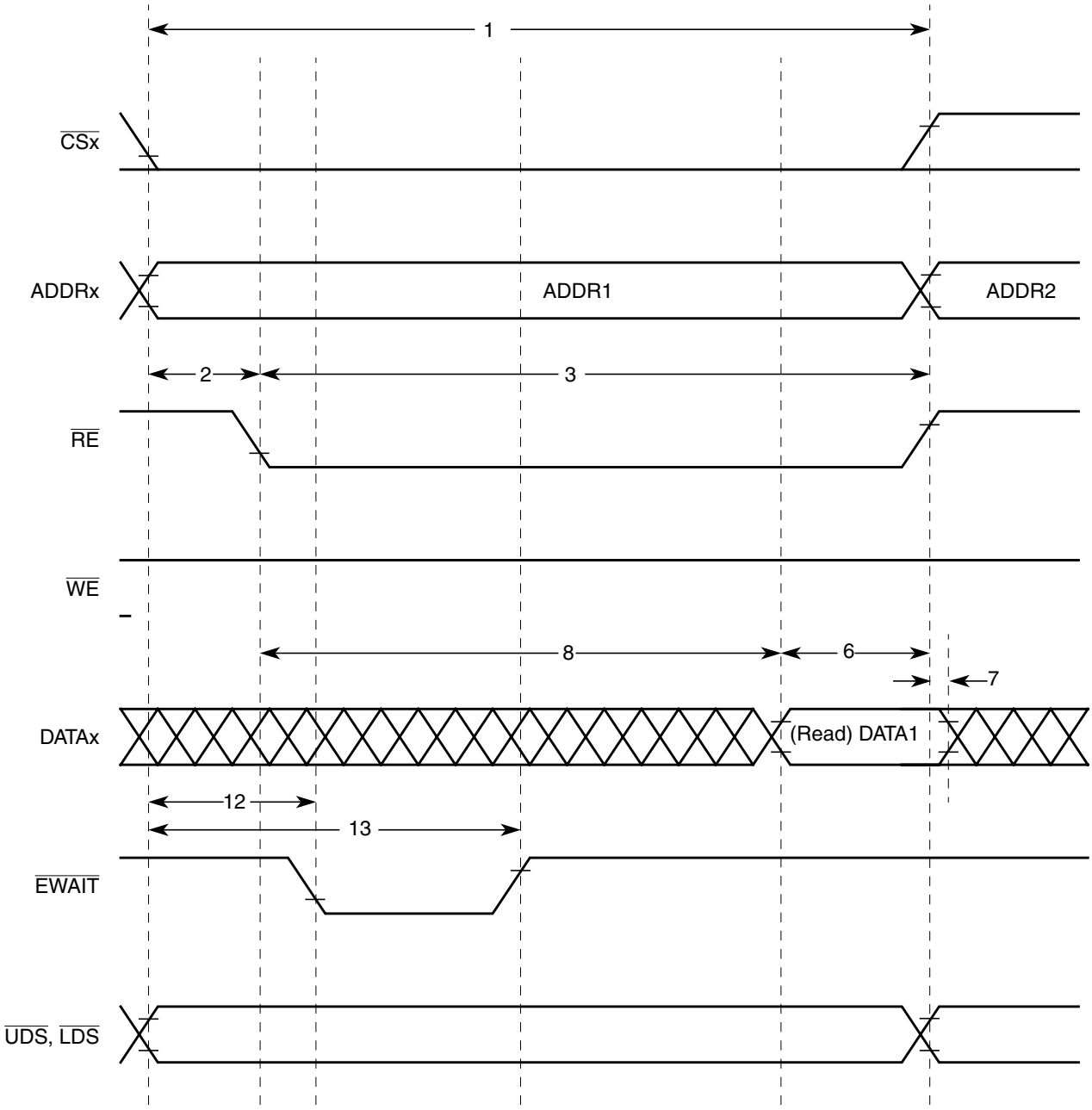



**Figure A-13. Example 1b: Normal Expanded Mode — Stretched Read Access**

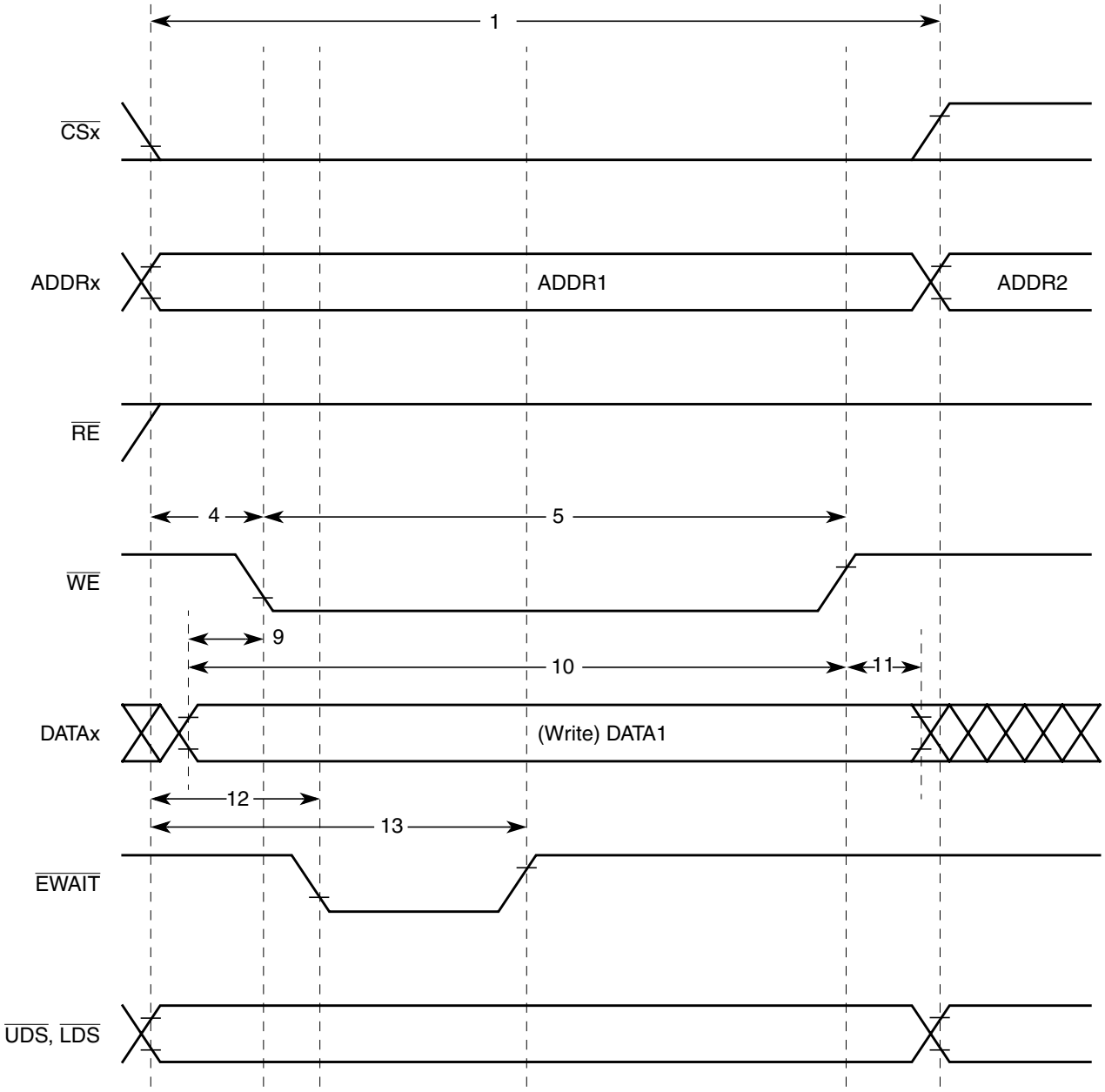



**Figure A-14. Example 1b: Normal Expanded Mode — Stretched Write Access**

**Table A-31. Example 1b: Normal Expanded Mode Timing at 50MHz bus (EWAIT enabled)**

| No. | Characteristic | Symbol | $V_{DD5}$ = 5.0V | | | | | $V_{DD5}$ = 3.3V | | | | | Unit |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | C | 2 stretch cycles | | 3 stretch cycles | | C | 2 stretch cycles | | 3 stretch cycles | | |
| | | | | Min | Max | Min | Max | | Min | Max | Min | Max | |
| - | Frequency of internal bus | $f_i$ | - | D.C. | 50.0 | D.C. | 50.0 | - | D.C. | 25.0 | D.C. | 25.0 | MHz |
| - | Internal cycle time | $t_{cyc}$ | - | 20 | ∞ | 20 | ∞ | - | 20 | ∞ | 20 | ∞ | ns |
| - | Frequency of external bus | $f_o$ | - | D.C. | 16.7 | D.C. | 12.5 | - | D.C. | 8.33 | D.C. | 6.25 | MHz |

**Table A-31. Example 1b: Normal Expanded Mode Timing at 50MHz bus (EWAIT enabled)**

| No. | Characteristic | Symbol | $V_{DD5}$ = 5.0V | | | | | $V_{DD5}$ = 3.3V | | | | | Unit |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | C | 2 stretch cycles | | 3 stretch cycles | | C | 2 stretch cycles | | 3 stretch cycles | | |
| | | | | Min | Max | Min | Max | | Min | Max | Min | Max | |
| - | External cycle time (selected by EXSTR) | $t_{cyce}$ | - | 60 | ∞ | 80 | ∞ | - | 120 | ∞ | 160 | ∞ | ns |
| 1 | External cycle time (EXSTR+1EWAIT) | $t_{cycew}$ | - | 80 | ∞ | 100 | ∞ | - | 160 | ∞ | 200 | ∞ | ns |
| 2 | Address (1) valid to $\overline{RE}$ fall | $t_{ADRE}$ | D | 4 | - | 4 | - | D | 13 | - | 13 | - | ns |
| 3 | Pulse width, $\overline{RE}$ (2) | $PW_{RE}$ | D | 68 | - | 88 | - | D | 138 | - | 178 | - | ns |
| 4 | Address valid to $\overline{WE}$ fall | $t_{ADWE}$ | D | 4 | - | 4 | - | D | 15 | - | 15 | - | ns |
| 5 | Pulse width, $\overline{WE}$ | $PW_{WE}$ | D | 58 | - | 78 | - | D | 118 | - | 158 | - | ns |
| 6 | Read data setup time (if ITHRS = 0) | $t_{DSR}$ | D | 19 | - | 19 | - | D | 38 | - | 38 | - | ns |
| | Read data setup time (if ITHRS = 1) | $t_{DSR}$ | D | 23 | - | 23 | - | D | N/A | | | | ns |
| 7 | Read data hold time | $t_{DHR}$ | D | 0 | - | 0 | - | D | 0 | - | 0 | - | ns |
| 8 | Read enable access time | $t_{ACCR}$ | D | 49 | - | 69 | - | D | 65 | - | 105 | - | ns |
| 9 | Write data valid to $\overline{WE}$ fall | $t_{WDWE}$ | D | 5 | - | 5 | - | D | 5 | - | 5 | - | ns |
| 10 | Write data setup time | $t_{DSW}$ | D | 63 | - | 93 | - | D | 123 | - | 163 | - | ns |
| 11 | Write data hold time | $t_{DHW}$ | D | 6 | - | 6 | - | D | 4 | - | 4 | - | ns |
| 12 | Address to $\overline{EWAIT}$ fall | $t_{ADWF}$ | D | 0 | 16 | 0 | 36 | D | 0 | 20 | 0 | - | ns |
| 13 | Address to $\overline{EWAIT}$ rise | $t_{ADWR}$ | D | 30 | 39 | 50 | 58 | D | 50 | 61 | 90 | 101 | ns |

1. Includes the following signals: ADDRx, $\overline{UDS}$, $\overline{LDS}$, and $\overline{CSx}$.
2. Affected by $\overline{EWAIT}$.

### A.7.3.3 Emulation Single-Chip Mode (Without Wait States)

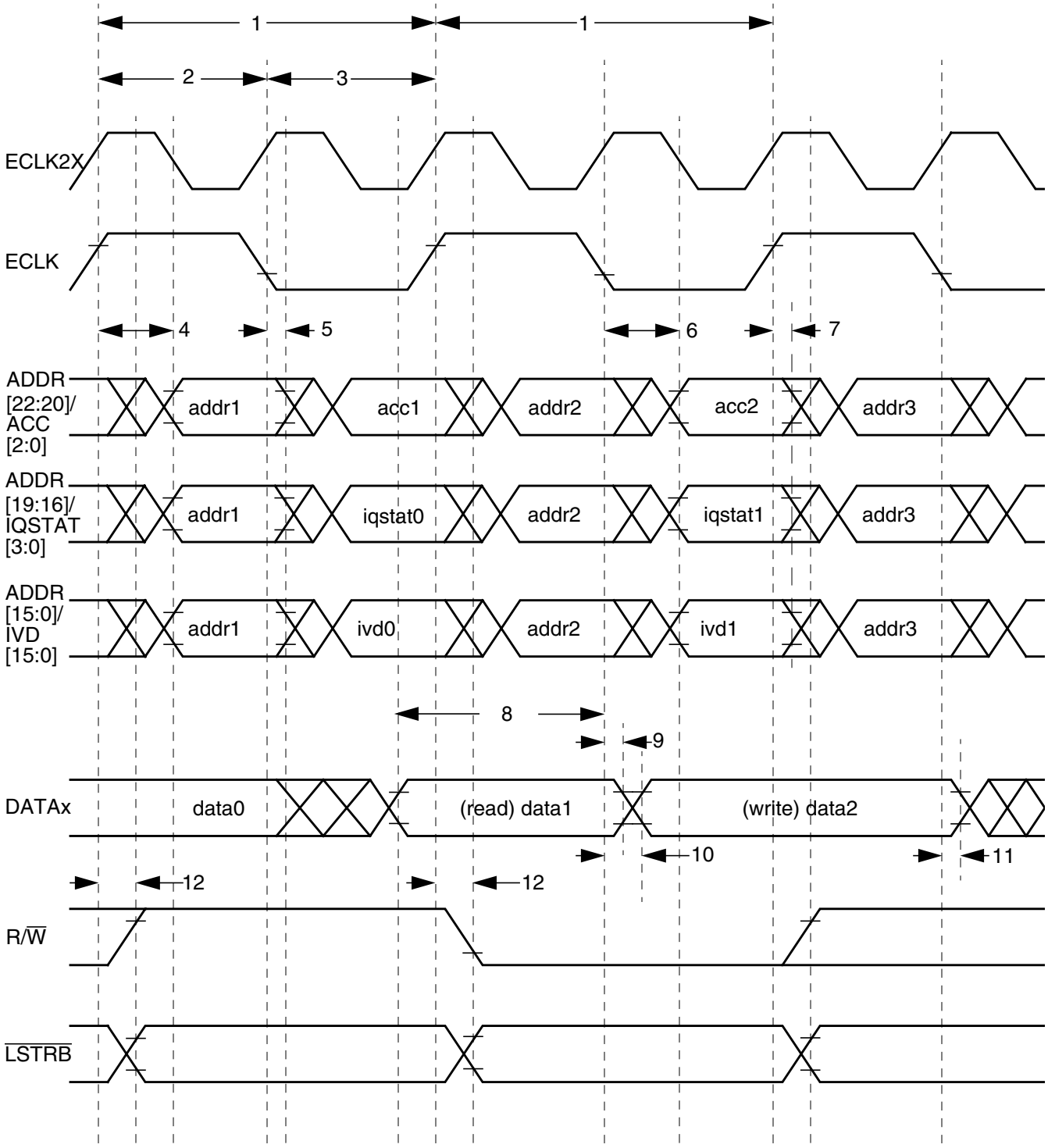



**Figure A-15. Example 2a: Emulation Single-Chip Mode — Read Followed by Write**

**Table A-32. Example 2a: Emulation Single-Chip Mode Timing 50 Mhz bus, $V_{DD5}$=5.0V (EWAIT disabled)**

| No. | C | Characteristic [1] | Symbol | Min | Max | Unit |
|---|---|---|---|---|---|---|
| - | - | Frequency of internal bus | $f_i$ | D.C. | 50.0 | MHz |
| 1 | - | Cycle time | $t_{cyc}$ | 20 | ∞ | ns |
| 2 | D | Pulse width, E high | $PW_{EH}$ | 9 | - | ns |
| 3 | D | Pulse width, E low | $PW_{EL}$ | 9 | - | ns |
| 4 | D | Address delay time | $t_{AD}$ | - | 5 | ns |
| 5 | D | Address hold time | $t_{AH}$ | 0 | - | ns |
| 6 | D | IVDx delay time [2] | $t_{IVDD}$ | - | 4.5 | ns |
| 7 | D | IVDx hold time | $t_{IVDH}$ | 0 | - | ns |
| 8 | D | Read data setup time (ITHRS = 1 only) | $t_{DSR}$ | 15 | - | ns |
| 9 | D | Read data hold time | $t_{DHR}$ | 0 | - | ns |
| 10 | D | Write data delay time | $t_{DDW}$ | - | 5 | ns |
| 11 | D | Write data hold time | $t_{DHW}$ | 0 | - | ns |
| 12 | D | Read/write data delay time [3] | $t_{RWD}$ | -1 | 5 | ns |

1. Typical Supply and Silicon, Room Temperature Only
2. Includes also ACCx, IQSTATx
3. Includes $\overline{\text{LSTRB}}$

### A.7.3.4 Emulation Expanded Mode (With Optional Access Stretching)

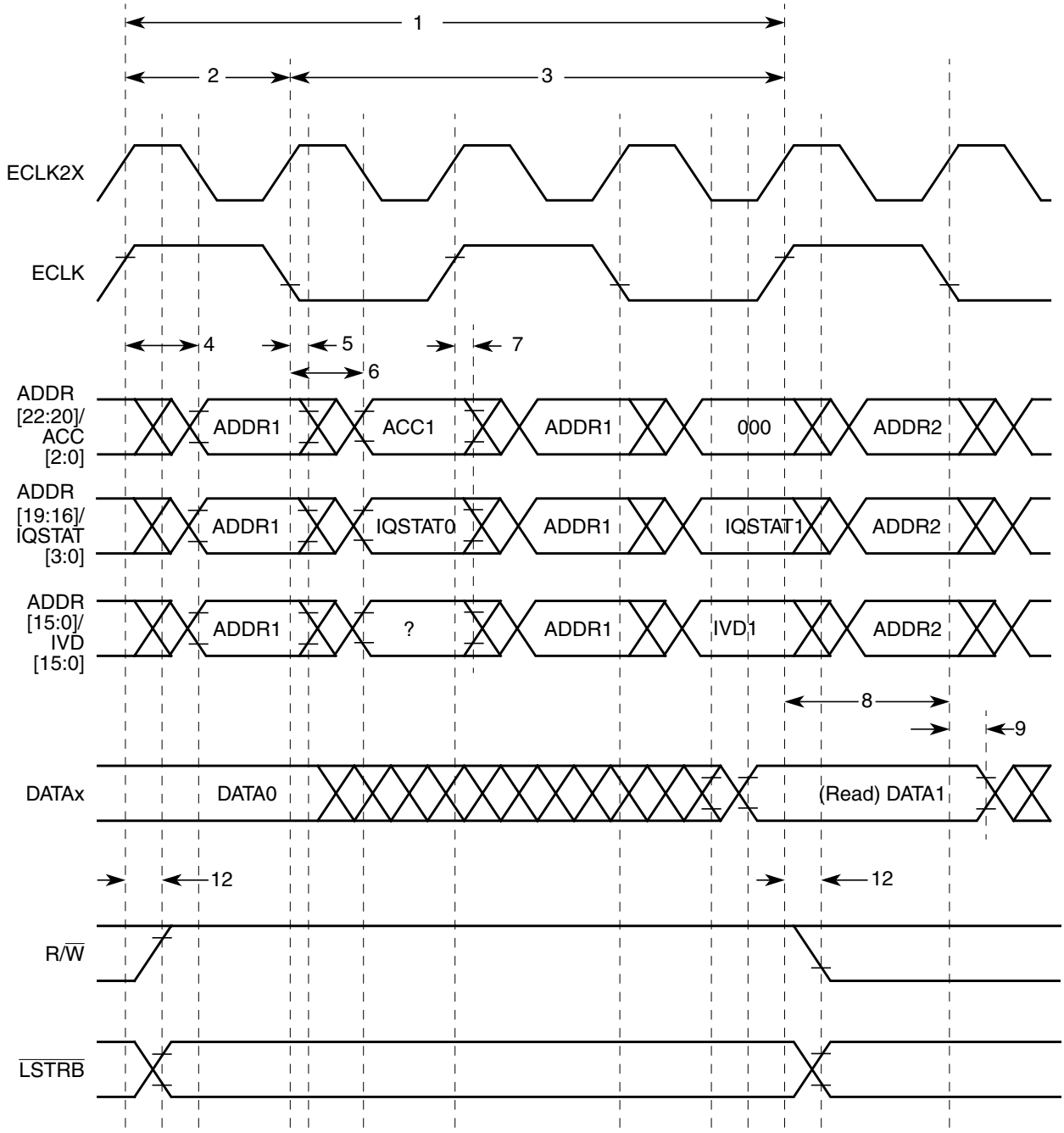



**Figure A-16. Example 2b: Emulation Expanded Mode — Read with 1 Stretch Cycle**

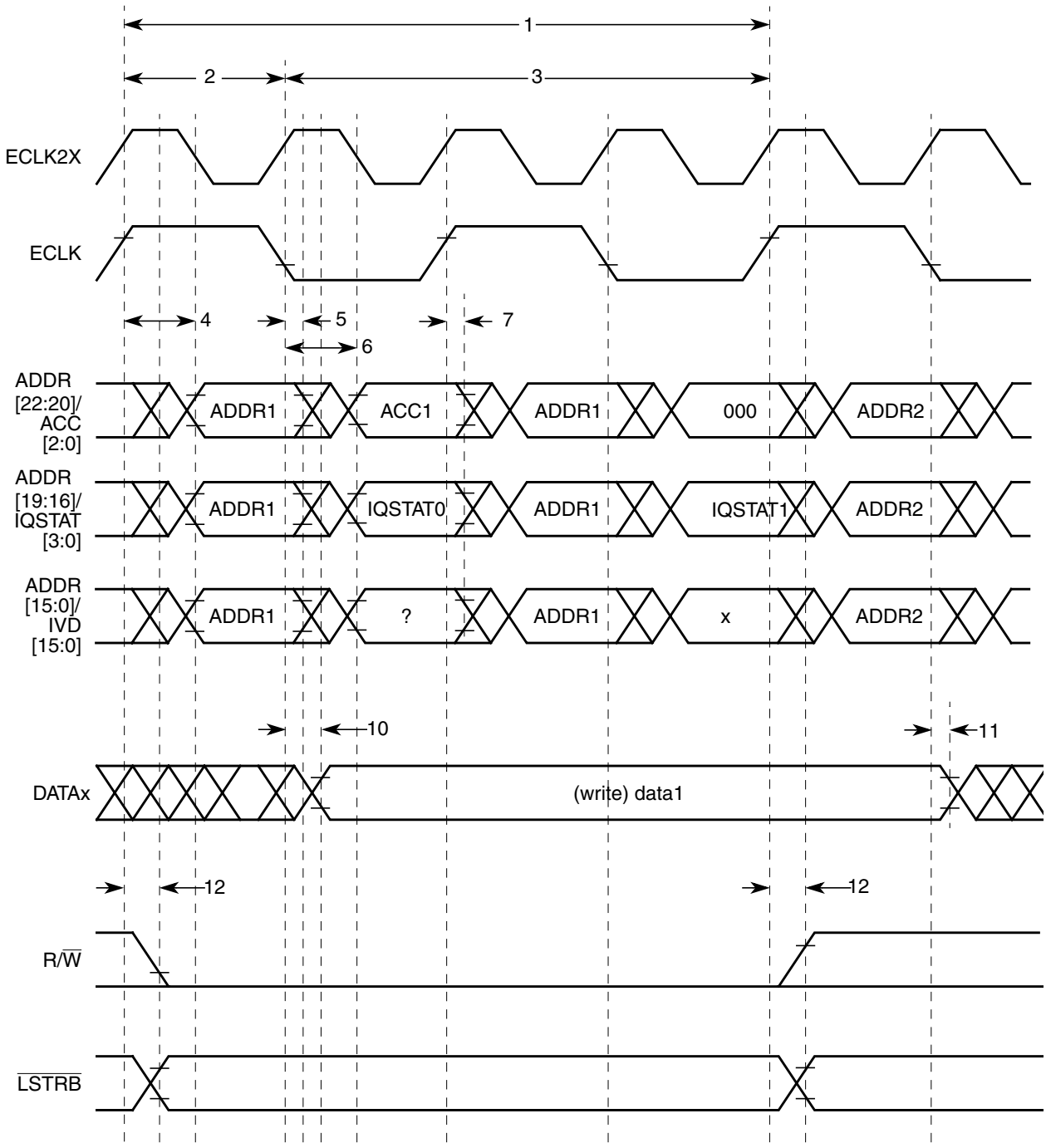



**Figure A-17. Example 2b: Emulation Expanded Mode Ò Write with 1 Stretch Cycle**

**Table A-33. Example 2b: Emulation Expanded Mode Timing 50 MHz bus, $V_{DD5}$=5.0V (EWAIT disabled)**

| No. | C | Characteristic [1] | Symbol | 1 stretch cycle | | 2 stretch cycles | | 3 stretch cycles | | Unit |
|---|---|---|---|---|---|---|---|---|---|---|
| | | | | Min | Max | Min | Max | Min | Max | |
| - | - | Internal cycle time | $t_{cyc}$ | 20 | ∞ | 20 | ∞ | 20 | ∞ | ns |
| 1 | - | Cycle time | $t_{cyce}$ | 40 | ∞ | 60 | ∞ | 80 | ∞ | ns |
| 2 | D | Pulse width, E high | $PW_{EH}$ | 9 | 11 | 9 | 11 | 9 | 11 | ns |
| 3 | D | E falling to sampling E rising | $t_{EFSR}$ | 28 | 32 | 48 | 52 | 68 | 72 | ns |
| 4 | D | Address delay time | $t_{AD}$ | refer to table Table A-32 | | refer to table Table A-32 | | refer to table Table A-32 | | ns |
| 5 | D | Address hold time | $t_{AH}$ | | | | | | | ns |
| 6 | D | IVD delay time [2] | $t_{IVDD}$ | | | | | | | ns |
| 7 | D | IVD hold time | $t_{IVDH}$ | | | | | | | ns |
| 8 | D | Read data setup time | $t_{DSR}$ | | | | | | | ns |
| 9 | D | Read data hold time | $t_{DHR}$ | | | | | | | ns |
| 10 | D | Write data delay time | $t_{DDW}$ | | | | | | | ns |
| 11 | D | Write data hold time | $t_{DHW}$ | | | | | | | ns |
| 12 | D | Read/write data delay time [3] | $t_{RWD}$ | | | | | | | ns |

1. Typical Supply and Silicon, Room Temperature Only
2. Includes also ACCx, IQSTATx
3. Includes LSTRB

### A.7.3.5 External Tag Trigger Timing

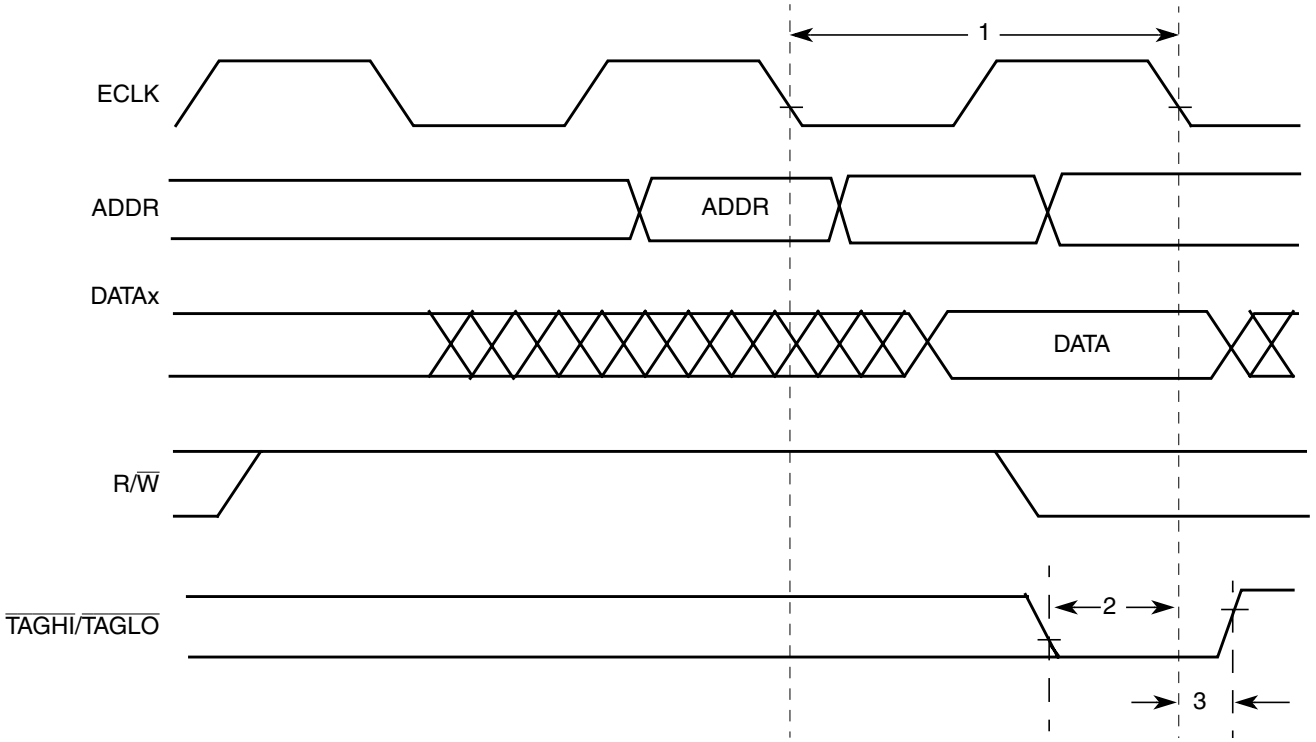



**Figure A-18. External Trigger Timing**

**Table A-34. External Tag Trigger Timing $V_{DD35}$ = 5.0 V**

| No. | C | Characteristic [1] | Symbol | Min | Max | Unit |
|---|---|---|---|---|---|---|
| - | D | Frequency of internal bus | $f_i$ | D.C. | 50.0 | MHz |
| 1 | D | Cycle time | $t_{cyc}$ | 20 | ∞ | ns |
| 2 | D | $\overline{TAGHI}/\overline{TAGLO}$ setup time | $t_{TS}$ | 10 | — | ns |
| 3 | D | $\overline{TAGHI}/\overline{TAGLO}$ hold time | $t_{TH}$ | 0 | — | ns |

1. Typical supply and silicon, room temperature only

# Appendix B
# Package Information

This section provides the physical dimensions of the packages.

# B.1 208 MAPBGA

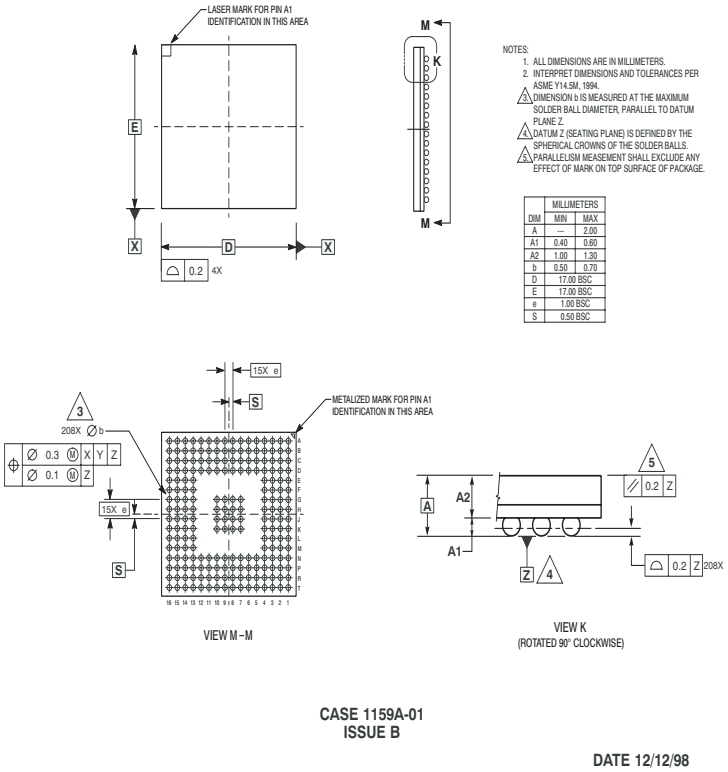
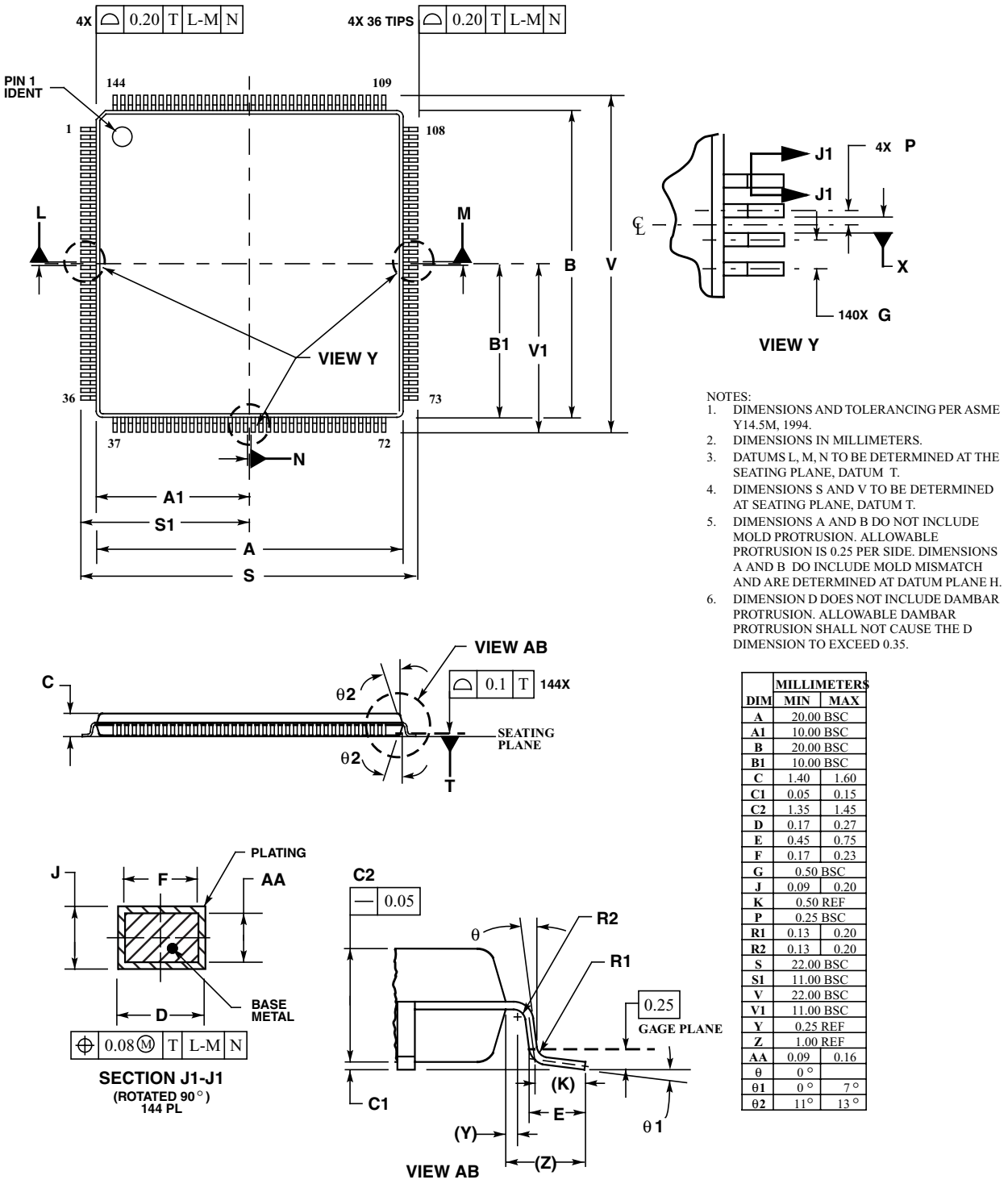



**Figure B-1. 208MAPBGA Mechanical Dimensions**

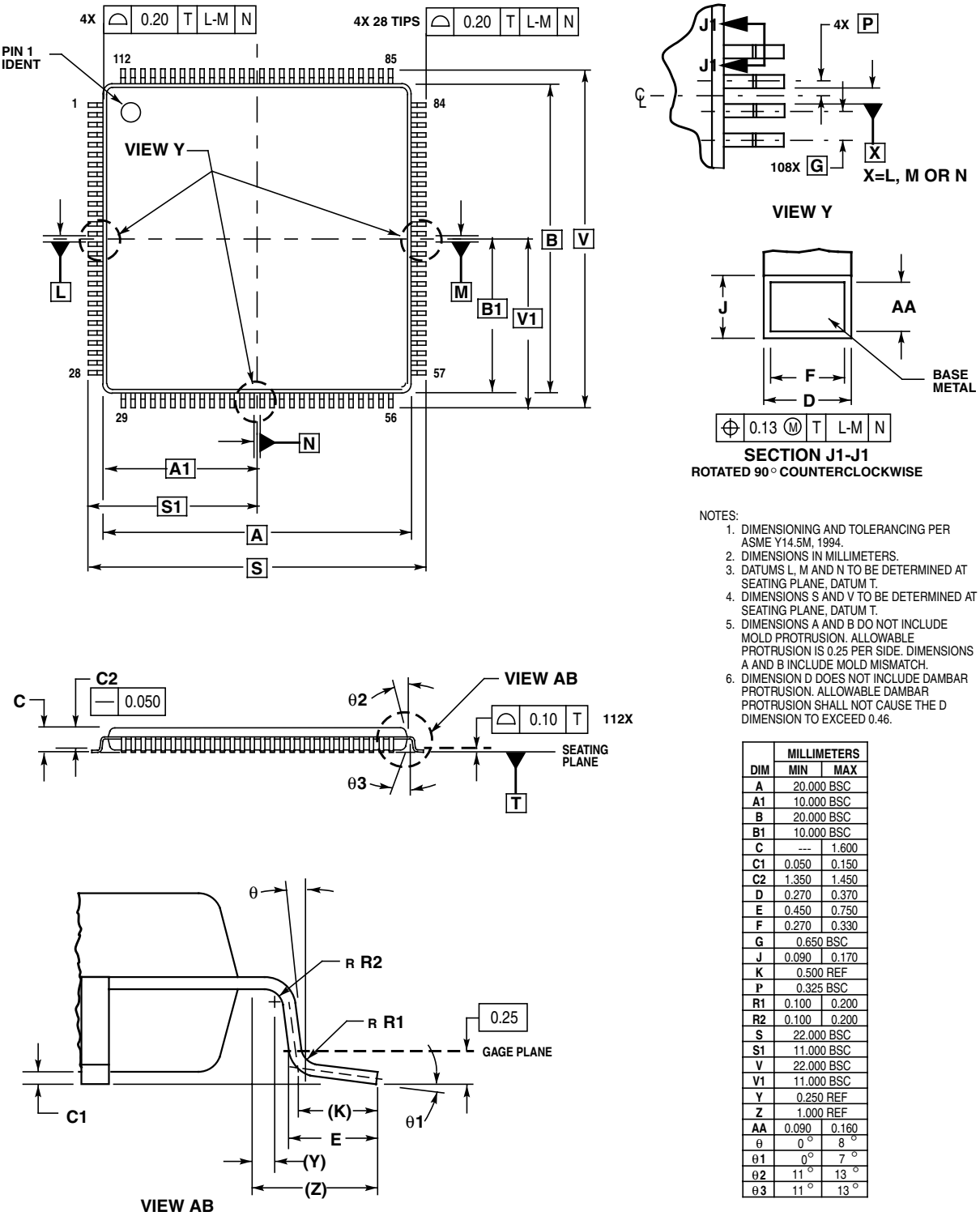
# B.2 144-Pin LQFP




| DIM | MILLIMETERS MIN | MAX |
|---|---|---|
| A | 20.00 BSC | |
| A1 | 10.00 BSC | |
| B | 20.00 BSC | |
| B1 | 10.00 BSC | |
| C | 1.40 | 1.60 |
| C1 | 0.05 | 0.15 |
| C2 | 1.35 | 1.45 |
| D | 0.17 | 0.27 |
| E | 0.45 | 0.75 |
| F | 0.17 | 0.23 |
| G | 0.50 BSC | |
| J | 0.09 | 0.20 |
| K | 0.50 REF | |
| P | 0.25 BSC | |
| R1 | 0.13 | 0.20 |
| R2 | 0.13 | 0.20 |
| S | 22.00 BSC | |
| S1 | 11.00 BSC | |
| V | 22.00 BSC | |
| V1 | 11.00 BSC | |
| Y | 0.25 REF | |
| Z | 1.00 REF | |
| AA | 0.09 | 0.16 |
| θ | 0 ° | |
| θ1 | 0 ° | 7 ° |
| θ2 | 11° | 13 ° |

**Figure B-2. 144-Pin LQFP Mechanical Dimensions (Case No. 918-03)**

# B.3 112-Pin LQFP Package

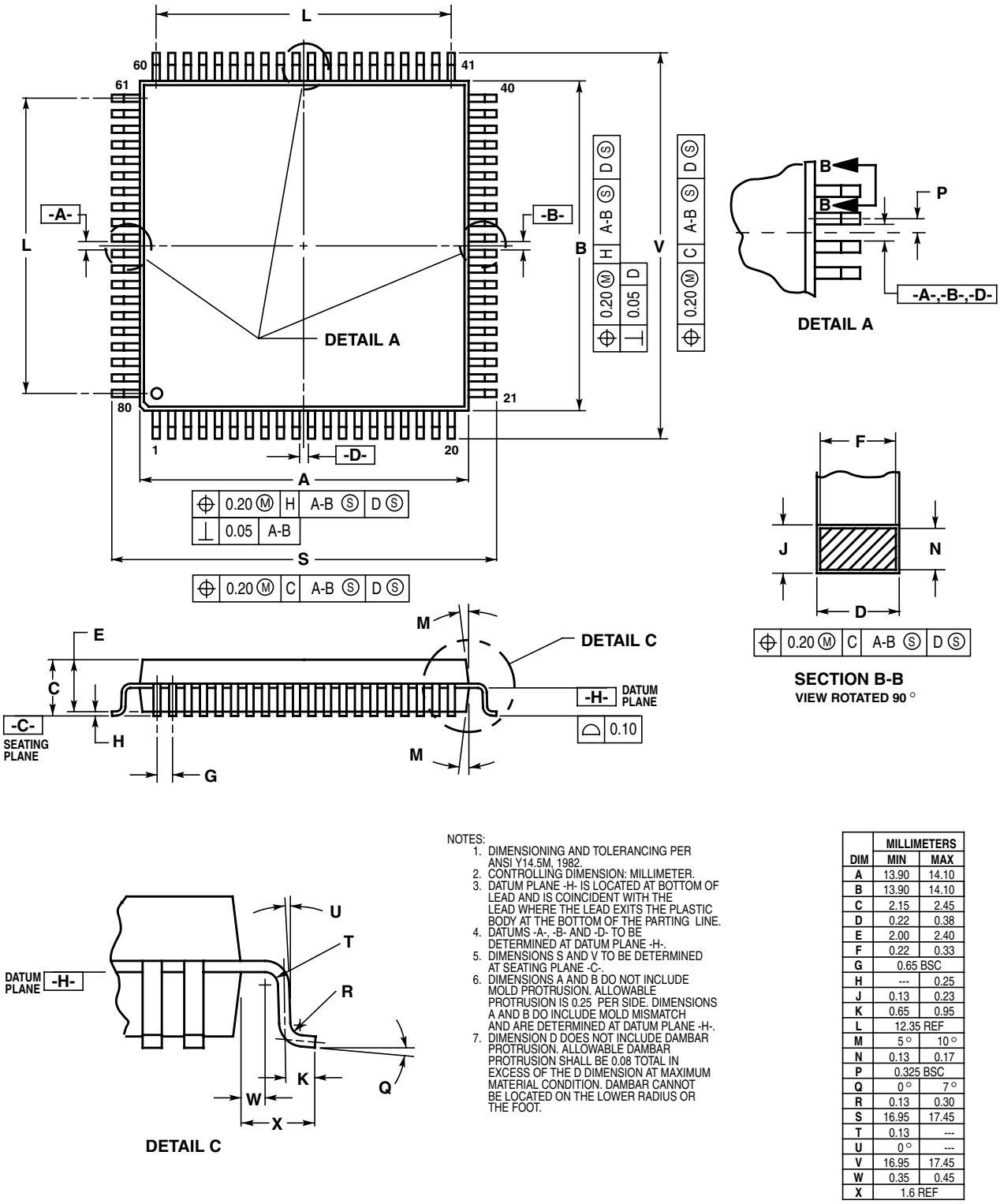



NOTES:
1. DIMENSIONING AND TOLERANCING PER ASME Y14.5M, 1994.
2. DIMENSIONS IN MILLIMETERS.
3. DATUMS L, M AND N TO BE DETERMINED AT SEATING PLANE, DATUM T.
4. DIMENSIONS S AND V TO BE DETERMINED AT SEATING PLANE, DATUM T.
5. DIMENSIONS A AND B DO NOT INCLUDE MOLD PROTRUSION. ALLOWABLE PROTRUSION IS 0.25 PER SIDE. DIMENSIONS A AND B INCLUDE MOLD MISMATCH.
6. DIMENSION D DOES NOT INCLUDE DAMBAR PROTRUSION. ALLOWABLE DAMBAR PROTRUSION SHALL NOT CAUSE THE D DIMENSION TO EXCEED 0.46.

| | MILLIMETERS | |
|---|---|---|
| DIM | MIN | MAX |
| A | 20.000 BSC | |
| A1 | 10.000 BSC | |
| B | 20.000 BSC | |
| B1 | 10.000 BSC | |
| C | --- | 1.600 |
| C1 | 0.050 | 0.150 |
| C2 | 1.350 | 1.450 |
| D | 0.270 | 0.370 |
| E | 0.450 | 0.750 |
| F | 0.270 | 0.330 |
| G | 0.650 BSC | |
| J | 0.090 | 0.170 |
| K | 0.500 REF | |
| P | 0.325 BSC | |
| R1 | 0.100 | 0.200 |
| R2 | 0.100 | 0.200 |
| S | 22.000 BSC | |
| S1 | 11.000 BSC | |
| V | 22.000 BSC | |
| V1 | 11.000 BSC | |
| Y | 0.250 REF | |
| Z | 1.000 REF | |
| AA | 0.090 | 0.160 |
| θ | 0° | 8° |
| θ1 | 0° | 7° |
| θ2 | 11° | 13° |
| θ3 | 11° | 13° |

**Figure B-3. 112-Pin LQFP Mechanical Dimensions (Case No. 987)**

# B.4 80-Pin QFP Package




NOTES:
1. DIMENSIONING AND TOLERANCING PER ANSI Y14.5M, 1982.
2. CONTROLLING DIMENSION: MILLIMETER.
3. DATUM PLANE -H- IS LOCATED AT BOTTOM OF LEAD AND IS COINCIDENT WITH THE LEAD WHERE THE LEAD EXITS THE PLASTIC BODY AT THE BOTTOM OF THE PARTING LINE.
4. DATUMS -A-, -B- AND -D- TO BE DETERMINED AT DATUM PLANE -H-.
5. DIMENSIONS S AND V TO BE DETERMINED AT SEATING PLANE -C-.
6. DIMENSIONS A AND B DO NOT INCLUDE MOLD PROTRUSION. ALLOWABLE PROTRUSION IS 0.25 PER SIDE. DIMENSIONS A AND B DO INCLUDE MOLD MISMATCH AND ARE DETERMINED AT DATUM PLANE -H-.
7. DIMENSION D DOES NOT INCLUDE DAMBAR PROTRUSION. ALLOWABLE DAMBAR PROTRUSION SHALL BE 0.08 TOTAL IN EXCESS OF THE D DIMENSION AT MAXIMUM MATERIAL CONDITION. DAMBAR CANNOT BE LOCATED ON THE LOWER RADIUS OR THE FOOT.

| | MILLIMETERS | |
|---|---|---|
| DIM | MIN | MAX |
| A | 13.90 | 14.10 |
| B | 13.90 | 14.10 |
| C | 2.15 | 2.45 |
| D | 0.22 | 0.38 |
| E | 2.00 | 2.40 |
| F | 0.22 | 0.33 |
| G | 0.65 BSC | |
| H | --- | 0.25 |
| J | 0.13 | 0.23 |
| K | 0.65 | 0.95 |
| L | 12.35 REF | |
| M | 5° | 10° |
| N | 0.13 | 0.17 |
| P | 0.325 BSC | |
| Q | 0° | 7° |
| R | 0.13 | 0.30 |
| S | 16.95 | 17.45 |
| T | 0.13 | --- |
| U | 0° | --- |
| V | 16.95 | 17.45 |
| W | 0.35 | 0.45 |
| X | 1.6 REF | |

**Figure B-4. 80-Pin QFP Mechanical Dimensions (Case No. 841B)**

# Appendix C
# PCB Layout Guidelines

The PCB must be carefully laid out to ensure proper operation of the voltage regulator as well as of the MCU itself. The following rules must be observed:

- Every supply pair must be decoupled by a ceramic capacitor connected as near as possible to the corresponding pins .
- Central point of the ground star should be the VSS3 pin.
- Use low ohmic low inductance connections between VSS1, VSS2 and VSS3.
- VSSPLL must be directly connected to VSS3.
- Keep traces of VSSPLL, EXTAL, and XTAL as short as possible and occupied board area for C7, C8, and Q1 as small as possible.
- Do not place other signals or supplies underneath area occupied by C7, C8, and Q1 and the connection area to the MCU.
- Central power input should be fed in at the VDDA/VSSA pins.

Example layouts are illustrated on the following pages.

**Table C-1. Recommended Decoupling Capacitor Choice**

| Component | Purpose | Type | Value |
|---|---|---|---|
| C1 | $V_{DDF}$ filter capacitor | Ceramic X7R | 220 nF |
| C2 | $V_{DDX4}$ filter capacitor (MAPBGA208, LQFP144 only | X7R/tantalum | >=100 nF |
| C3 | $V_{DDX2}$ filter capacitor | X7R/tantalum | >=100 nF |
| C4 | $V_{DDPLL}$ filter capacitor | Ceramic X7R | 220 nF |
| C5 | OSC load capacitor | From crystal manufacturer | |
| C6 | OSC load capacitor | | |
| C7 | $V_{DDR}$ filter capacitor | X7R/tantalum | >=100 nF |
| C8 | $V_{DDX3}$ filter capacitor (MAPBGA208, LQFP144 only | X7R/tantalum | >=100 nF |
| C9 | $V_{DD}$ filter capacitor | Ceramic X7R | 220 nF |
| C10 | $V_{DDA1}$ filter capacitor | Ceramic X7R | >=100 nF |
| C11 | $V_{DDX1}$ filter capacitor | X7R/tantalum | >=100 nF |
| C12 | $V_{DDX5}$ filter capacitor (MAPBGA208 package only) | X7R/tantalum | >=100 nF |
| C13 | $V_{DDX6}$ filter capacitor (MAPBGA208 package only) | X7R/tantalum | >=100 nF |
| C14 | $V_{DDX7}$ filter capacitor (MAPBGA208 package only) | X7R/tantalum | >=100 nF |
| C15 | $V_{DDA2}$ filter capacitor(MAPBGA208 package only) | Ceramic X7R | >=100 nF |
| Q1 | Quartz | — | — |

Figure C-1. 144-Pin LQFP Recommended PCB Layout (Loop Controlled Pierce Oscillator)

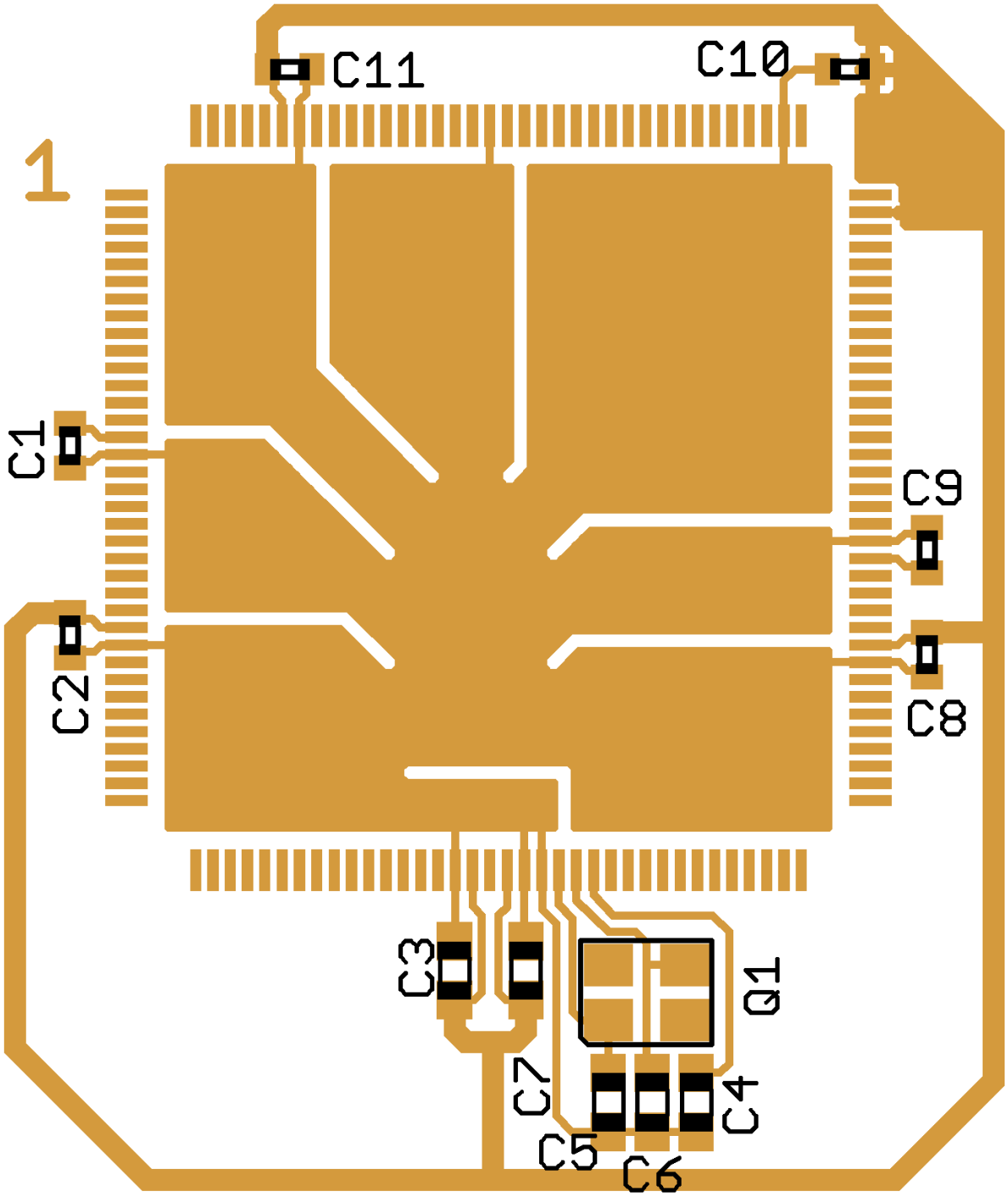


**Figure C-2. 112-Pin LQFP Recommended PCB Layout (Loop Controlled Pierce Oscillator)**

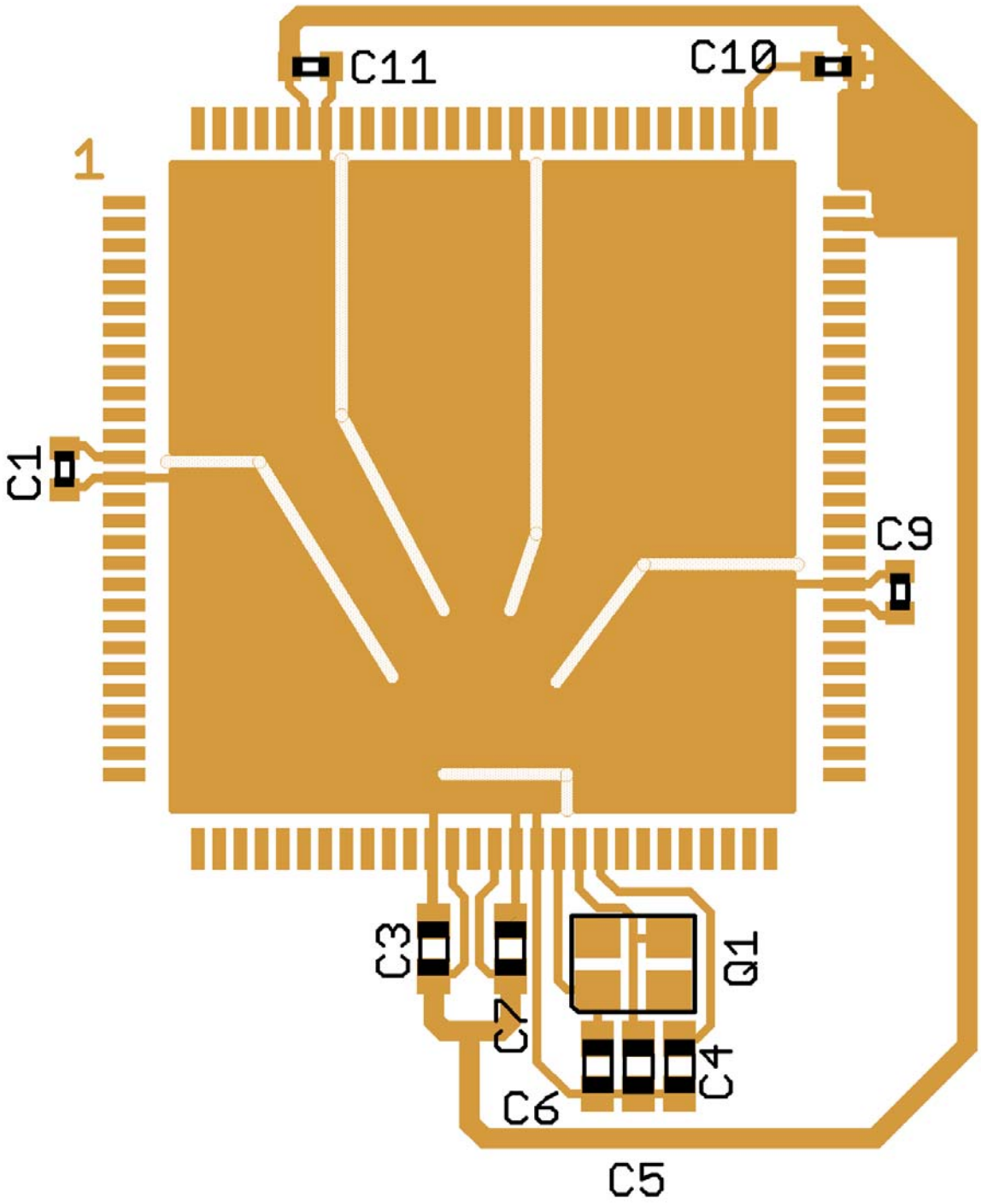
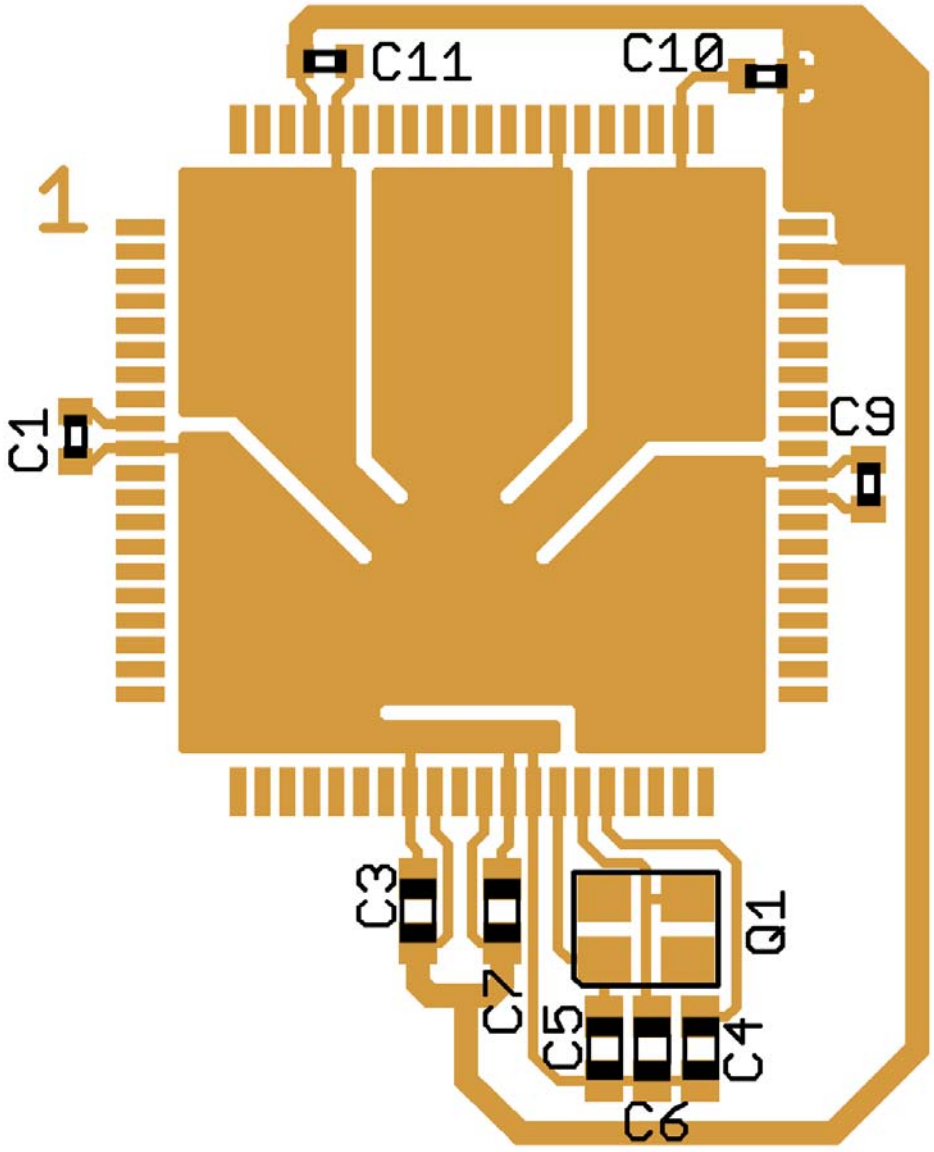


**Figure C-3. 80-Pin QFP Recommended PCB Layout (Loop Controlled Pierce Oscillator)**



# Appendix D
# Derivative Differences

## D.1 Memory Sizes and Package Options S12XE - Family

**Table D-1. Package and Memory Options of MC9S12XE-Family**

| Device | Package | Flash | RAM | EEPROM | D-Flash |
|---|---|---|---|---|---|
| 9S12XEP100 | 208 MAPBGA | 1M | 64K | 4K | 32K |
| | 144 LQFP | | | | |
| | 112 LQFP | | | | |
| 9S12XEP768 | 208 MAPBGA | 768K | 48K | | |
| | 144 LQFP | | | | |
| | 112 LQFP | | | | |
| 9S12XEQ512 | 144 LQFP | 512K | 32K | | |
| | 112 LQFP | | | | |
| | 80 QFP | | | | |
| 9S12XEQ384<br>9S12XEG384 | 144 LQFP | 384K | 24K | | |
| | 112 LQFP | | | | |
| | 80 QFP | | | | |
| 9S12XES384 | 144 LQFP | 384K | 16K | | |
| | 112 LQFP | | | | |
| | 80 QFP | | | | |
| 9S12XET256<br>9S12XEA256(1) | 144 LQFP | 256K | 16K | | |
| | 112 LQFP | | | | |
| | 80 QFP | | | | |
| 9S12XEG256 | 112 LQFP | | | | |
| 9S12XEG128<br>9S12XEA128[1] | 112 LQFP | 128K | 12K | 2K | 32K |
| | 80 QFP | | | | |

1. The 9S12XEA128/256 are a special bondouts for access to extra ADC channels in 80QFP.Available in 80QFP / 256K/128K memory sizes only. WARNING: NOT PIN-COMPATIBLE WITH REST OF FAMILY. The 9S12XET256/9S12XEG128 use the standard 80QFP bondout, compatible with other family members.

**Table D-2. Peripheral Options of MC9S12XE-Family Members**

| Device | Package | XGATE | CAN | SCI | SPI | IIC | ECT | TIM[1] | PIT | A/D | I/O |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 9S12XEP100 | 208 MAPBGA | yes | 5 | 8 | 3 | 2 | 8ch | 8ch | 8ch | 2/32 | 152 |
| | 144LQFP | | 5 | 8 | 3 | 2 | 8ch | 8ch | 8ch | 2/24 | 119 |
| | 112LQFP | | 5 | 8 | 3 | 1 | 8ch | 8ch | 8ch | 2/16[2] | 91 |
| 9S12XEP768 | 208 MAPBGA | | 5 | 8 | 3 | 2 | 8ch | 8ch | 8ch | 2/32 | 152 |
| | 144LQFP | | 5 | 8 | 3 | 2 | 8ch | 8ch | 8ch | 2/24 | 119 |
| | 112LQFP | | 5 | 8 | 3 | 1 | 8ch | 8ch | 8ch | 2/16 (2) | 91 |
| 9S12XEQ512 | 144LQFP | | 4 | 6 | 3 | 2 | 8ch | 0 | 4ch | 2/24 | 119 |
| | 112LQFP | | 4 | 6 | 3 | 1 | 8ch | 0 | 4ch | 2/16[2] | 91 |
| | 80QFP | | 4 | 2 | 3 | 1 | 8ch | 0 | 4ch | 2/8[2] | 59 |
| 9S12XEQ384 | 144LQFP | | 4 | 6 | 3 | 2 | 8ch | 0 | 4ch | 2/24 | 119 |
| | 112LQFP | | 4 | 6 | 3 | 1 | 8ch | 0 | 4ch | 2/16[2] | 91 |
| | 80QFP | | 4 | 2 | 3 | 1 | 8ch | 0 | 4ch | 2/8[2] | 59 |
| 9S12XEG384 | 144LQFP | | 2 | 6 | 3 | 2 | 8ch | 0 | 4ch | 2/24 | 119 |
| | 112LQFP | | 2 | 6 | 3 | 1 | 8ch | 0 | 4ch | 2/16[2] | 91 |
| | 80QFP | | 2 | 2 | 3 | 1 | 8ch | 0 | 4ch | 2/8[2] | 59 |
| 9S12XES384 | 144LQFP | no | 1 | 2 | 1 | 2 | 8ch | 0 | 4ch | 2/24 | 119 |
| | 112LQFP | | 1 | 2 | 1 | 1 | 8ch | 0 | 4ch | 2/16[2] | 91 |
| | 80QFP | | 1 | 2 | 1 | 1 | 8ch | 0 | 4ch | 2/8[2] | 59 |
| 9S12XET256 | 144LQFP | yes | 3 | 4 | 3 | 1 | 8ch | 0 | 4ch | 2/24 | 119 |
| | 112LQFP | | 3 | 4 | 3 | 1 | 8ch | 0 | 4ch | 2/16[2] | 91 |
| | 80QFP | | 3 | 2 | 3 | 1 | 8ch | 0 | 4ch | 2/8[2] | 59 |
| 9S12XEA256 (3) | 80QFP | | 3 | 2 | 3 | 1 | 8ch | 0 | 4ch | 2/12[2] | 59 |
| 9S12XEG256 | 112LQFP | | 2 | 4 | 3 | 1 | 8ch | 0 | 4ch | 2/16[2] | 91 |
| 9S12XEG128 | 112LQFP | | 2 | 2 | 2 | 1 | 8ch | 0 | 2ch | 1/16 | 91 |
| | 80QFP | | 2 | 2 | 2 | 1 | 8ch | 0 | 2ch | 1/8 | 59 |
| 9S12XEA128[3] | 80QFP | | 2 | 2 | 2 | 1 | 8ch | 0 | 2ch | 2/12[2] | 59 |

1. TIM available via rerouting on EP100,EP768 devices 112/144 pinout options.
   TIM not available on EG128,ET256,EA256, EQ384,EQ512 devices.
2. The device features 2 16-channel ATD modules, only one of which is bonded out in this package option
3. This is a special bondout for access to extra ADC channels in 80QFP.
   WARNING: NOT PIN-COMPATIBLE WITH REST OF FAMILY.
   The 9S12XET256/9S12XEG128 use the standard 80QFP bondouts, compatible with other family members.

## D.2 Pinout explanations:

- Pinout compatibility is maintained throughout the device family
- A/D is the number of modules/total number of A/D channels.
- I/O is the sum of ports capable to act as digital input or output. .
- For additional flexibility especially for the low pin count packages several I/O functions can be routed under software control to different pins. For details refer to the device overview section..

- Versions with 5 CAN modules will have CAN0, CAN1, CAN2, CAN3 and CAN4.
- Versions with 4 CAN modules will have CAN0, CAN1, CAN2 and CAN4.
- Versions with 3 CAN modules will have CAN0, CAN1 and CAN4.
- Versions with 2 CAN modules will have CAN0 and CAN4.
- Versions with 1 CAN module will have CAN0.

- Versions with 3 SPI modules will have SPI0, SPI1 and SPI2.
- Versions with 2 SPI modules will have SPI0 and SPI1.
- Versions with 1 SPI modules will have SPI0.

- Versions with 8 SCI modules will have SCI0, SCI1, SCI2, SCI3, SCI4, SCI5, SCI6 and SCI7.
- Versions with 7 SCI modules will have SCI0, SCI1, SCI2, SCI3, SCI4, SCI5, and SCI6.
- Versions with 6 SCI modules will have SCI0, SCI1, SCI2, SCI3, SCI4 and SCI5.
- Versions with 5 SCI modules will have SCI0, SCI1, SCI2, SCI3 and SCI4.
- Versions with 4 SCI modules will have SCI0, SCI1, SCI2 and SCI4.
- Versions with 3 SCI modules will have SCI0, SCI1 and SCI2.
- Versions with 2 SCI modules will have SCI0 and SCI1.
- Versions with 1 SCI module will have SCI0.

- Versions with 2 IIC modules will have IIC0 and IIC1.
- Versions with 1 IIC module will have IIC0.

- Versions with 1 ATD module will have ATD0.

# Appendix E
# Detailed Register Address Map

The following tables show the detailed register map of the S12XE-Family.

**NOTE**

Smaller derivatives within the S12XE-Family feature a subset of the listed modules. Refer to Appendix D Derivative Differences for more information about derivative device module subsets.

## 0x0000–0x0009 Port Integration Module (PIM) Map 1 of 6

| Address | Name | | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|---|---|
| 0x0000 | PORTA | R | PA7 | PA6 | PA5 | PA4 | PA3 | PA2 | PA1 | PA 0 |
| | | W | | | | | | | | |
| 0x0001 | PORTB | R | PB7 | PB6 | PB5 | PB4 | PB3 | PB2 | PB1 | PB0 |
| | | W | | | | | | | | |
| 0x0002 | DDRA | R | DDRA7 | DDRA6 | DDRA5 | DDRA4 | DDRA3 | DDRA2 | DDRA1 | DDRA0 |
| | | W | | | | | | | | |
| 0x0003 | DDRB | R | DDRB7 | DDRB6 | DDRB5 | DDRB4 | DDRB3 | DDRB2 | DDRB1 | DDRB0 |
| | | W | | | | | | | | |
| 0x0004 | PORTC | R | PC7 | PC6 | PC5 | PC4 | PC3 | PC2 | PC1 | PC0 |
| | | W | | | | | | | | |
| 0x0005 | PORTD | R | PD7 | PD6 | PD5 | PD4 | PD3 | PD2 | PD1 | PD0 |
| | | W | | | | | | | | |
| 0x0006 | DDRC | R | DDRC7 | DDRC6 | DDRC5 | DDRC4 | DDRC3 | DDRC2 | DDRC1 | DDRC0 |
| | | W | | | | | | | | |
| 0x0007 | DDRD | R | DDRD7 | DDRD6 | DDRD5 | DDRD4 | DDRD3 | DDRD2 | DDRD1 | DDRD0 |
| | | W | | | | | | | | |
| 0x0008 | PORTE | R | PE7 | PE6 | PE5 | PE4 | PE3 | PE2 | PE1 | PE0 |
| | | W | | | | | | | | |
| 0x0009 | DDRE | R | DDRE7 | DDRE6 | DDRE5 | DDRE4 | DDRE3 | DDRE2 | 0 | 0 |
| | | W | | | | | | | | |

## 0x000A–0x000B Module Mapping Control (S12XMMC) Map 1 of 2

| Address | Name | | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|---|---|
| 0x000A | MMCCTL0 | R | CS3E1 | CS2E1 | CS1E1 | CS0E1 | CS3E0 | CS2E0 | CS1E0 | CS0E0 |
| | | W | | | | | | | | |
| 0x000B | MODE | R | MODC | MODB | MODA | 0 | 0 | 0 | 0 | 0 |
| | | W | | | | | | | | |

## 0x000C–0x000D Port Integration Module (PIM) Map 2 of 6

| Address | Name | | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|---|---|
| 0x000C | PUCR | R | PUPKE | BKPUE | 0 | PUPEE | PUPDE | PUPCE | PUPBE | PUPAE |
| | | W | | | | | | | | |
| 0x000D | RDRIV | R | RDPK | 0 | 0 | RDPE | RDPD | RDPC | RDPB | RDPA |
| | | W | | | | | | | | |

## 0x000E–0x000F External Bus Interface (S12XEBI) Map

| Address | Name | | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|---|---|
| 0x000E | EBICTL0 | R | ITHRS | 0 | HDBE | ASIZ4 | ASIZ3 | ASIZ2 | ASIZ1 | ASIZ0 |
| | | W | | | | | | | | |
| 0x000F | EBICTL1 | R | 0 | EXSTR12 | EXSTR11 | EXSTR10 | 0 | EXSTR02 | EXSTR01 | EXSTR00 |
| | | W | | | | | | | | |

## 0x0010–0x0017 Module Mapping Control (S12XMMC) Map 2 of 2

| Address | Name | | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|---|---|
| 0x0010 | GPAGE | R | 0 | GP6 | GP5 | GP4 | GP3 | GP2 | GP1 | GP0 |
| | | W | | | | | | | | |
| 0x0011 | DIRECT | R | DP15 | DP14 | DP13 | DP12 | DP11 | DP10 | DP9 | DP8 |
| | | W | | | | | | | | |
| 0x0012 | Reserved | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | W | | | | | | | | |
| 0x0013 | MMCCTL1 | R | TGMRAMON | MGROMON | EEEIFRON | PGMIFRON | RAMHM | EROMON | ROMHM | ROMON |
| | | W | | | | | | | | |
| 0x0014 | Reserved | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | W | | | | | | | | |
| 0x0015 | PPAGE | R | PIX7 | PIX6 | PIX5 | PIX4 | PIX3 | PIX2 | PIX1 | PIX0 |
| | | W | | | | | | | | |
| 0x0016 | RPAGE | R | RP7 | RP6 | RP5 | RP4 | RP3 | RP2 | RP1 | RP0 |
| | | W | | | | | | | | |
| 0x0017 | EPAGE | R | EP7 | EP6 | EP5 | EP4 | EP3 | EP2 | EP1 | EP0 |
| | | W | | | | | | | | |

## 0x0018–0x001B Miscellaneous Peripheral

| Address | Name | | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|---|---|
| 0x0018 | Reserved | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | W | | | | | | | | |
| 0x0019 | Reserved | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | W | | | | | | | | |
| 0x001A | PARTIDH(1) | R | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 0 |
| | | W | | | | | | | | |
| 0x001B | PARTIDL[1] | R | 1 | 0 | 0 | 1 | 0 | 0 | 1 | 0 |
| | | W | | | | | | | | |

1. Refer to Part ID assignments in the device description section for a full list of S12XE-FamilyPart ID values.

## 0x001C–0x001D Port Integration Module (PIM) Map 3 of 6

| Address | Name | | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|---|---|
| 0x001C | ECLKCTL | R | NECLK | NCLKX2 | DIV16 | EDIV4 | EDIV3 | EDIV2 | EDIV1 | EDIV0 |
| | | W | | | | | | | | |
| 0x001D | Reserved | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | W | | | | | | | | |

## 0x001E–0x001F Port Integration Module (PIM) Map 3 of 6

| Address | Name | | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|---|---|
| 0x001E | IRQCR | R | IRQE | IRQEN | 0 | 0 | 0 | 0 | 0 | 0 |
| | | W | | | | | | | | |
| 0x001F | Reserved | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | W | | | | | | | | |

## 0x0020–0x0027 Debug Module (S12XDBG) Map

| Address | Name | | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|---|---|
| 0x0020 | DBGC1 | R | ARM | 0 | XGSBPE | BDM | DBGBRK | | COMRV | |
| | | W | | TRIG | | | | | | |
| 0x0021 | DBGSR | R | TBF | EXTF | 0 | 0 | 0 | SSF2 | SSF1 | SSF0 |
| | | W | | | | | | | | |
| 0x0022 | DBGTCR | R | TSOURCE | | TRANGE | | TRCMOD | | TALIGN | |
| | | W | | | | | | | | |
| 0x0023 | DBGC2 | R | 0 | 0 | 0 | 0 | CDCM | | ABCM | |
| | | W | | | | | | | | |
| 0x0024 | DBGTBH | R | Bit 15 | Bit 14 | Bit 13 | Bit 12 | Bit 11 | Bit 10 | Bit 9 | Bit 8 |
| | | W | | | | | | | | |
| 0x0025 | DBGTBL | R | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
| | | W | | | | | | | | |
| 0x0026 | DBGCNT | R | 0 | CNT | | | | | | |
| | | W | | | | | | | | |
| 0x0027 | DBGSCRX | R | 0 | 0 | 0 | 0 | SC3 | SC2 | SC1 | SC0 |
| | | W | | | | | | | | |
| 0x0027 | DBGMFR | R | 0 | 0 | 0 | 0 | MC3 | MC2 | MC1 | MC0 |
| | | W | | | | | | | | |
| 0x0028 (1) | DBGXCTL (COMPA/C) | R | 0 | NDB | TAG | BRK | RW | RWE | SRC | COMPE |
| | | W | | | | | | | | |
| 0x0028 (2) | DBGXCTL (COMPB/D) | R | SZE | SZ | TAG | BRK | RW | RWE | SRC | COMPE |
| | | W | | | | | | | | |
| 0x0029 | DBGXAH | R | 0 | Bit 22 | 21 | 20 | 19 | 18 | 17 | Bit 16 |
| | | W | | | | | | | | |
| 0x002A | DBGXAM | R | Bit 15 | 14 | 13 | 12 | 11 | 10 | 9 | Bit 8 |
| | | W | | | | | | | | |
| 0x002B | DBGXAL | R | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
| | | W | | | | | | | | |
| 0x002C | DBGXDH | R | Bit 15 | 14 | 13 | 12 | 11 | 10 | 9 | Bit 8 |
| | | W | | | | | | | | |
| 0x002D | DBGXDL | R | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
| | | W | | | | | | | | |
| 0x002E | DBGXDHM | R | Bit 15 | 14 | 13 | 12 | 11 | 10 | 9 | Bit 8 |
| | | W | | | | | | | | |
| 0x002F | DBGXDLM | R | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
| | | W | | | | | | | | |

1. This represents the contents if the Comparator A or C control register is blended into this address
2. This represents the contents if the Comparator B or D control register is blended into this address

## 0x0030–0x0031 Reserved Register Space

| | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| 0x0030 | Reserved | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | W | | | | | | | | |
| 0x0031 | Reserved | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | W | | | | | | | | |

## 0x0032–0x0033 Port Integration Module (PIM) Map 4 of 6

| Address | Name | | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|---|---|
| 0x0032 | PORTK | R<br>W | PK7 | PK6 | PK5 | PK4 | PK3 | PK2 | PK1 | PK0 |
| 0x0033 | DDRK | R<br>W | DDRK7 | DDRK6 | DDRK5 | DDRK4 | DDRK3 | DDRK2 | DDRK1 | DDRK0 |

## 0x0034–0x003F Clock and Reset Generator (CRG) Map

| Address | Name | | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|---|---|
| 0x0034 | SYNR | R<br>W | VCOFRQ[1:0] | | SYNDIV5 | SYNDIV4 | SYNDIV3 | SYNDIV2 | SYNDIV1 | SYNDIV0 |
| 0x0035 | REFDV | R<br>W | REFFRQ[1:0] | | REFDIV5 | REFDIV4 | REFDIV3 | REFDIV2 | REFDIV1 | REFDIV0 |
| 0x0036 | POSTDIV | R<br>W | 0 | 0 | 0 | POSTDIV[4:0]] | | | | |
| 0x0037 | CRGFLG | R<br>W | RTIF | PORF | LVRF | LOCKIF | LOCK | ILAF | SCMIF | SCM |
| 0x0038 | CRGINT | R<br>W | RTIE | 0 | 0 | LOCKIE | 0 | 0 | SCMIE | 0 |
| 0x0039 | CLKSEL | R<br>W | PLLSEL | PSTP | XCLKS | 0 | PLLWAI | 0 | RTIWAI | COPWAI |
| 0x003A | PLLCTL | R<br>W | CME | PLLON | FM1 | FM0 | FSTWKP | PRE | PCE | SCME |
| 0x003B | RTICTL | R<br>W | RTDEC | RTR6 | RTR5 | RTR4 | RTR3 | RTR2 | RTR1 | RTR0 |
| 0x003C | COPCTL | R<br>W | WCOP | RSBCK | 0<br>WRTMASK | 0 | 0 | CR2 | CR1 | CR0 |
| 0x003D | FORBYP | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | W | Reserved For Factory Test | | | | | | | |
| 0x003E | CTCTL | R | 0 | 0 | 0 | 0 | | 0 | 0 | 0 |
| | | W | Reserved For Factory Test | | | | | | | |
| 0x003F | ARMCOP | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | W | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |

## 0x0040–0x007F Enhanced Capture Timer 16-Bit 8-Channels (ECT) Map (Sheet 1 of 3)

| Address | Name | | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|---|---|
| 0x0040 | TIOS | R | IOS7 | IOS6 | IOS5 | IOS4 | IOS3 | IOS2 | IOS1 | IOS0 |
| | | W | | | | | | | | |
| 0x0041 | CFORC | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | W | FOC7 | FOC6 | FOC5 | FOC4 | FOC3 | FOC2 | FOC1 | FOC0 |
| 0x0042 | OC7M | R | OC7M7 | OC7M6 | OC7M5 | OC7M4 | OC7M3 | OC7M2 | OC7M1 | OC7M0 |
| | | W | | | | | | | | |
| 0x0043 | OC7D | R | OC7D7 | OC7D6 | OC7D5 | OC7D4 | OC7D3 | OC7D2 | OC7D1 | OC7D0 |
| | | W | | | | | | | | |
| 0x0044 | TCNT (high) | R | TCNT15 | TCNT14 | TCNT13 | TCNT12 | TCNT11 | TCNT10 | TCNT9 | TCNT8 |
| | | W | | | | | | | | |
| 0x0045 | TCNT (low) | R | TCNT7 | TCNT6 | TCNT5 | TCNT4 | TCNT3 | TCNT2 | TCNT1 | TCNT0 |
| | | W | | | | | | | | |
| 0x0046 | TSCR1 | R | TEN | TSWAI | TSFRZ | TFFCA | PRNT | 0 | 0 | 0 |
| | | W | | | | | | | | |
| 0x0047 | TTOV | R | TOV7 | TOV6 | TOV5 | TOV4 | TOV3 | TOV2 | TOV1 | TOV0 |
| | | W | | | | | | | | |
| 0x0048 | TCTL1 | R | OM7 | OL7 | OM6 | OL6 | OM5 | OL5 | OM4 | OL4 |
| | | W | | | | | | | | |
| 0x0049 | TCTL2 | R | OM3 | OL3 | OM2 | OL2 | OM1 | OL1 | OM0 | OL0 |
| | | W | | | | | | | | |
| 0x004A | TCTL3 | R | EDG7B | EDG7A | EDG6B | EDG6A | EDG5B | EDG5A | EDG4B | EDG4A |
| | | W | | | | | | | | |
| 0x004B | TCTL4 | R | EDG3B | EDG3A | EDG2B | EDG2A | EDG1B | EDG1A | EDG0B | EDG0A |
| | | W | | | | | | | | |
| 0x004C | TIE | R | C7I | C6I | C5I | C4I | C3I | C2I | C1I | C0I |
| | | W | | | | | | | | |
| 0x004D | TSCR2 | R | TOI | 0 | 0 | 0 | TCRE | PR2 | PR1 | PR0 |
| | | W | | | | | | | | |
| 0x004E | TFLG1 | R | C7F | C6F | C5F | C4F | C3F | C2F | C1F | C0F |
| | | W | | | | | | | | |
| 0x004F | TFLG2 | R | TOF | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | W | | | | | | | | |
| 0x0050 | TC0 (hi) | R | Bit 15 | 14 | 13 | 12 | 11 | 10 | 9 | Bit 8 |
| | | W | | | | | | | | |
| 0x0051 | TC0 (lo) | R | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
| | | W | | | | | | | | |
| 0x0052 | TC1 (hi) | R | Bit 15 | 14 | 13 | 12 | 11 | 10 | 9 | Bit 8 |
| | | W | | | | | | | | |
| 0x0053 | TC1 (lo) | R | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
| | | W | | | | | | | | |
| 0x0054 | TC2 (hi) | R | Bit 15 | 14 | 13 | 12 | 11 | 10 | 9 | Bit 8 |
| | | W | | | | | | | | |
| 0x0055 | TC2 (lo) | R | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
| | | W | | | | | | | | |

## 0x0040–0x007F Enhanced Capture Timer 16-Bit 8-Channels (ECT) Map (Sheet 2 of 3)

| Address | Name | | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|---|---|
| 0x0056 | TC3 (hi) | R<br>W | Bit 15 | 14 | 13 | 12 | 11 | 10 | 9 | Bit 8 |
| 0x0057 | TC3 (lo) | R<br>W | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
| 0x0058 | TC4 (hi) | R<br>W | Bit 15 | 14 | 13 | 12 | 11 | 10 | 9 | Bit 8 |
| 0x0059 | TC4 (lo) | R<br>W | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
| 0x005A | TC5 (hi) | R<br>W | Bit 15 | 14 | 13 | 12 | 11 | 10 | 9 | Bit 8 |
| 0x005B | TC5 (lo) | R<br>W | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
| 0x005C | TC6 (hi) | R<br>W | Bit 15 | 14 | 13 | 12 | 11 | 10 | 9 | Bit 8 |
| 0x005D | TC6 (lo) | R<br>W | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
| 0x005E | TC7 (hi) | R<br>W | Bit 15 | 14 | 13 | 12 | 11 | 10 | 9 | Bit 8 |
| 0x005F | TC7 (lo) | R<br>W | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
| 0x0060 | PACTL | R<br>W | 0 | PAEN | PAMOD | PEDGE | CLK1 | CLK0 | PAOVI | PAI |
| 0x0061 | PAFLG | R<br>W | 0 | 0 | 0 | 0 | 0 | 0 | PAOVF | PAIF |
| 0x0062 | PACN3 (hi) | R<br>W | PACNT7 (15) | PACNT6 (14) | PACNT5 (13) | PACNT4 (12) | PACNT3 (11) | PACNT2 (10) | PACNT1 (9) | PACNT0 (8) |
| 0x0063 | PACN2 (lo) | R<br>W | PACNT7 | PACNT6 | PACNT5 | PACNT4 | PACNT3 | PACNT2 | PACNT1 | PACNT0 |
| 0x0064 | PACN1 (hi) | R<br>W | PACNT7 (15) | PACNT6 (14) | PACNT5 (13) | PACNT4 (12) | PACNT3 (11) | PACNT2 (10) | PACNT1 (9) | PACNT0 (8) |
| 0x0065 | PACN0 (lo) | R<br>W | PACNT7 | PACNT6 | PACNT5 | PACNT4 | PACNT3 | PACNT2 | PACNT1 | PACNT0 |
| 0x0066 | MCCTL | R<br>W | MCZI | MODMC | RDMCL | 0<br>ICLAT | 0<br>FLMC | MCEN | MCPR1 | MCPR0 |
| 0x0067 | MCFLG | R<br>W | MCZF | 0 | 0 | 0 | POLF3 | POLF2 | POLF1 | POLF0 |
| 0x0068 | ICPAR | R<br>W | 0 | 0 | 0 | 0 | PA3EN | PA2EN | PA1EN | PA0EN |
| 0x0069 | DLYCT | R<br>W | DLY7 | DLY6 | DLY5 | DLY4 | DLY3 | DLY2 | DLY1 | DLY0 |
| 0x006A | ICOVW | R<br>W | NOVW7 | NOVW6 | NOVW5 | NOVW4 | NOVW3 | NOVW2 | NOVW1 | NOVW0 |
| 0x006B | ICSYS | R<br>W | SH37 | SH26 | SH15 | SH04 | TFMOD | PACMX | BUFEN | LATQ |
| 0x006C | OCPD | R<br>W | OCPD7 | OCPD6 | OCPD5 | OCPD4 | OCPD3 | OCPD2 | OCPD1 | OCPD0 |

## 0x0040–0x007F Enhanced Capture Timer 16-Bit 8-Channels (ECT) Map (Sheet 3 of 3)

| Address | Name | | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|---|---|
| 0x006D | TIMTST | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | W | Reserved For Factory Test | | | | | | | |
| 0x006E | PTPSR | R | PTPS7 | PTPS6 | PTPS5 | PTPS4 | PTPS3 | PTPS2 | PTPS1 | PTPS0 |
| | | W | | | | | | | | |
| 0x006F | PTMCPSR | R | PTMPS7 | PTMPS6 | PTMPS5 | PTMPS4 | PTMPS3 | PTMPS2 | PTMPS1 | PTMPS0 |
| | | W | | | | | | | | |
| 0x0070 | PBCTL | R | 0 | PBEN | 0 | 0 | 0 | 0 | PBOVI | 0 |
| | | W | | | | | | | | |
| 0x0071 | PBFLG | R | 0 | 0 | 0 | 0 | 0 | 0 | PBOVF | 0 |
| | | W | | | | | | | | |
| 0x0072 | PA3H | R | PA3H7 | PA3H6 | PA3H5 | PA3H4 | PA3H3 | PA3H2 | PA3H1 | PA3H0 |
| | | W | | | | | | | | |
| 0x0073 | PA2H | R | PA2H7 | PA2H6 | PA2H5 | PA2H4 | PA2H3 | PA2H2 | PA2H1 | PA2H0 |
| | | W | | | | | | | | |
| 0x0074 | PA1H | R | PA1H7 | PA1H6 | PA1H5 | PA1H4 | PA1H3 | PA1H2 | PA1H1 | PA1H 0 |
| | | W | | | | | | | | |
| 0x0075 | PA0H | R | PA0H7 | PA0H6 | PA0H5 | PA0H4 | PA0H3 | PA0H2 | PA0H1 | PA0H0 |
| | | W | | | | | | | | |
| 0x0076 | MCCNT (hi) | R | MCCNT15 | MCCNT14 | MCCNT13 | MCCNT12 | MCCNT11 | MCCNT10 | MCCNT9 | MCCNT8 |
| | | W | | | | | | | | |
| 0x0077 | MCCNT (lo) | R | MCCNT7 | MCCNT6 | MCCNT5 | MCCNT4 | MCCNT3 | MCCNT2 | MCCNT1 | MCCNT0 |
| | | W | | | | | | | | |
| 0x0078 | TC0H (hi) | R | TC15 | TC14 | TC13 | TC12 | TC11 | TC10 | TC9 | TC8 |
| | | W | | | | | | | | |
| 0x0079 | TC0H (lo) | R | TC7 | TC6 | TC5 | TC4 | TC3 | TC2 | TC1 | TC0 |
| | | W | | | | | | | | |
| 0x007A | TC1H (hi) | R | TC15 | TC14 | TC13 | TC12 | TC11 | TC10 | TC9 | TC8 |
| | | W | | | | | | | | |
| 0x007B | TC1H (lo) | R | TC7 | TC6 | TC5 | TC4 | TC3 | TC2 | TC1 | TC0 |
| | | W | | | | | | | | |
| 0x007C | TC2H (hi) | R | TC15 | TC14 | TC13 | TC12 | TC11 | TC10 | TC9 | TC8 |
| | | W | | | | | | | | |
| 0x007D | TC2H (lo) | R | TC7 | TC6 | TC5 | TC4 | TC3 | TC2 | TC1 | TC0 |
| | | W | | | | | | | | |
| 0x007E | TC3H (hi) | R | TC15 | TC14 | TC13 | TC12 | TC11 | TC10 | TC9 | TC8 |
| | | W | | | | | | | | |
| 0x007F | TC3H (lo) | R | TC7 | TC6 | TC5 | TC4 | TC3 | TC2 | TC1 | TC0 |
| | | W | | | | | | | | |

## 0x0080–0x00AF Analog-to-Digital Converter 12-bit 16-Channels (ATD1) Map (Sheet 1 of 3)

| Address | Name | | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|---|---|
| 0x0080 | ATD1CTL0 | R | 0 | 0 | 0 | 0 | WRAP3 | WRAP2 | WRAP1 | WRAP0 |
| | | W | | | | | | | | |
| 0x0081 | ATD1CTL1 | R | ETRIG SEL | SRES1 | SRES0 | SMP_DIS | ETRIG CH3 | ETRIG CH2 | ETRIG CH1 | ETRIG CH0 |
| | | W | | | | | | | | |
| 0x0082 | ATD1CTL2 | R | 0 | AFFC | ICLKSTP | ETRIGLE | ETRIGP | ETRIGE | ASCIE | ACMPIE |
| | | W | | | | | | | | |
| 0x0083 | ATD1CTL3 | R | DJM | S8C | S4C | S2C | S1C | FIFO | FRZ1 | FRZ0 |
| | | W | | | | | | | | |
| 0x0084 | ATD1CTL4 | R | SMP2 | SMP1 | SMP0 | PRS4 | PRS3 | PRS2 | PRS1 | PRS0 |
| | | W | | | | | | | | |
| 0x0085 | ATD1CTL5 | R | 0 | SC | SCAN | MULT | CD | CC | CB | CA |
| | | W | | | | | | | | |
| 0x0086 | ATD1STAT0 | R | SCF | 0 | ETORF | FIFOR | CC3 | CC2 | CC1 | CC0 |
| | | W | | | | | | | | |
| 0x0087 | Reserved | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | W | | | | | | | | |
| 0x0088 | ATD1CMPEH | R | CMPE15 | CMPE14 | CMPE13 | CMPE12 | CMPE11 | CMPE10 | CMPE9 | CMPE8 |
| | | W | | | | | | | | |
| 0x0089 | ATD1CMPEL | R | CMPE7 | CMPE6 | CMPE5 | CMPE4 | CMPE3 | CMPE2 | CMPE1 | CMPE0 |
| | | W | | | | | | | | |
| 0x008A | ATD1STAT2H | R | CCF15 | CCF14 | CCF13 | CCF12 | CCF11 | CCF10 | CCF9 | CCF8 |
| | | W | | | | | | | | |
| 0x008B | ATD1STATL | R | CCF7 | CCF6 | CCF5 | CCF4 | CCF3 | CCF2 | CCF1 | CCF0 |
| | | W | | | | | | | | |
| 0x008C | ATD1DIENH | R | IEN15 | IEN14 | IEN13 | IEN12 | IEN11 | IEN10 | IEN9 | IEN8 |
| | | W | | | | | | | | |
| 0x008D | ATD1DIENL | R | IEN7 | IEN6 | IEN5 | IEN4 | IEN3 | IEN2 | IEN1 | IEN0 |
| | | W | | | | | | | | |
| 0x008E | ATD1CMPHTH | R | CMPHT15 | CMPHT14 | CMPHT13 | CMPHT12 | CMPHT11 | CMPHT10 | CMPHT9 | CMPHT8 |
| | | W | | | | | | | | |
| 0x008F | ATD1CMPHTL | R | CMPHT7 | CMPHT6 | CMPHT5 | CMPHT4 | CMPHT3 | CMPHT2 | CMPHT1 | CMPHT0 |
| | | W | | | | | | | | |
| 0x0090 | ATD1DR0H | R | Bit15 | 14 | 13 | 12 | 11 | 10 | 9 | Bit8 |
| | | W | | | | | | | | |
| 0x0091 | ATD1DR0L | R | Bit7 | Bit6 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | W | | | | | | | | |
| 0x0092 | ATD1DR1H | R | Bit15 | 14 | 13 | 12 | 11 | 10 | 9 | Bit8 |
| | | W | | | | | | | | |
| 0x0093 | ATD1DR1L | R | Bit7 | Bit6 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | W | | | | | | | | |
| 0x0094 | ATD1DR2H | R | Bit15 | 14 | 13 | 12 | 11 | 10 | 9 | Bit8 |
| | | W | | | | | | | | |
| 0x0095 | ATD1DR2L | R | Bit7 | Bit6 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | W | | | | | | | | |

## 0x0080–0x00AF Analog-to-Digital Converter 12-bit 16-Channels (ATD1) Map (Sheet 2 of 3)

| Address | Name | | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|---|---|
| 0x0096 | ATD1DR3H | R | Bit15 | 14 | 13 | 12 | 11 | 10 | 9 | Bit8 |
| | | W | | | | | | | | |
| 0x0097 | ATD1DR3L | R | Bit7 | Bit6 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | W | | | | | | | | |
| 0x0098 | ATD1DR4H | R | Bit15 | 14 | 13 | 12 | 11 | 10 | 9 | Bit8 |
| | | W | | | | | | | | |
| 0x0099 | ATD1DR4L | R | Bit7 | Bit6 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | W | | | | | | | | |
| 0x009A | ATD1DR5H | R | Bit15 | 14 | 13 | 12 | 11 | 10 | 9 | Bit8 |
| | | W | | | | | | | | |
| 0x009B | ATD1DR5L | R | Bit7 | Bit6 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | W | | | | | | | | |
| 0x009C | ATD1DR6H | R | Bit15 | 14 | 13 | 12 | 11 | 10 | 9 | Bit8 |
| | | W | | | | | | | | |
| 0x009D | ATD1DR6L | R | Bit7 | Bit6 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | W | | | | | | | | |
| 0x009E | ATD1DR7H | R | Bit15 | 14 | 13 | 12 | 11 | 10 | 9 | Bit8 |
| | | W | | | | | | | | |
| 0x009F | ATD1DR7L | R | Bit7 | Bit6 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | W | | | | | | | | |
| 0x00A0 | ATD1DR8H | R | Bit15 | 14 | 13 | 12 | 11 | 10 | 9 | Bit8 |
| | | W | | | | | | | | |
| 0x00A1 | ATD1DR8L | R | Bit7 | Bit6 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | W | | | | | | | | |
| 0x00A2 | ATD1DR9H | R | Bit15 | 14 | 13 | 12 | 11 | 10 | 9 | Bit8 |
| | | W | | | | | | | | |
| 0x00A3 | ATD1DR9L | R | Bit7 | Bit6 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | W | | | | | | | | |
| 0x00A4 | ATD1DR10H | R | Bit15 | 14 | 13 | 12 | 11 | 10 | 9 | Bit8 |
| | | W | | | | | | | | |
| 0x00A5 | ATD1DR10L | R | Bit7 | Bit6 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | W | | | | | | | | |
| 0x00A6 | ATD1DR11H | R | Bit15 | 14 | 13 | 12 | 11 | 10 | 9 | Bit8 |
| | | W | | | | | | | | |
| 0x00A7 | ATD1DR11L | R | Bit7 | Bit6 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | W | | | | | | | | |
| 0x00A8 | ATD1DR12H | R | Bit15 | 14 | 13 | 12 | 11 | 10 | 9 | Bit8 |
| | | W | | | | | | | | |
| 0x00A9 | ATD1DR12L | R | Bit7 | Bit6 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | W | | | | | | | | |
| 0x00AA | ATD1DR13H | R | Bit15 | 14 | 13 | 12 | 11 | 10 | 9 | Bit8 |
| | | W | | | | | | | | |
| 0x00AB | ATD1DR13L | R | Bit7 | Bit6 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | W | | | | | | | | |

## 0x0080–0x00AF Analog-to-Digital Converter 12-bit 16-Channels (ATD1) Map (Sheet 3 of 3)

| Address | Name | | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|---|---|
| 0x00AC | ATD1DR14H | R | Bit15 | 14 | 13 | 12 | 11 | 10 | 9 | Bit8 |
| | | W | | | | | | | | |
| 0x00AD | ATD1DR14L | R | Bit7 | Bit6 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | W | | | | | | | | |
| 0x00AE | ATD1DR15H | R | Bit15 | 14 | 13 | 12 | 11 | 10 | 9 | Bit8 |
| | | W | | | | | | | | |
| 0x00AF | ATD1DR15L | R | Bit7 | Bit6 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | W | | | | | | | | |

## 0x00B0–0x00B7 Inter IC Bus (IIC1) Map

| Address | Name | | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|---|---|
| 0x00B0 | IBAD | R | ADR7 | ADR6 | ADR5 | ADR4 | ADR3 | ADR2 | ADR1 | 0 |
| | | W | | | | | | | | |
| 0x00B1 | IBFD | R | IBC7 | IBC6 | IBC5 | IBC4 | IBC3 | IBC2 | IBC1 | IBC0 |
| | | W | | | | | | | | |
| 0x00B2 | IBCR | R | IBEN | IBIE | MS/$\overline{SL}$ | TX/$\overline{RX}$ | TXAK | 0 | 0 | IBSWAI |
| | | W | | | | | | RSTA | | |
| 0x00B3 | IBSR | R | TCF | IAAS | IBB | IBAL | 0 | SRW | IBIF | RXAK |
| | | W | | | | | | | | |
| 0x00B4 | IBDR | R | D7 | D6 | D5 | D4 | D3 | D2 | D1 | D 0 |
| | | W | | | | | | | | |
| 0x00B5 | IBCR2 | R | GCEN | ADTYPE | 0 | 0 | 0 | ADR10 | ADR9 | ADR8 |
| | | W | | | | | | | | |
| 0x00B6 | Reserved | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | W | | | | | | | | |
| 0x00B7 | Reserved | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | W | | | | | | | | |

## 0x00B8–0x00BF Asynchronous Serial Interface (SCI2) Map

| Address | Name | | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|---|---|
| 0x00B8 | SCI2BDH[(1)] | R | IREN | TNP1 | TNP0 | SBR12 | SBR11 | SBR10 | SBR9 | SBR8 |
| | | W | | | | | | | | |
| 0x00B9 | SCI2BDL[1] | R | SBR7 | SBR6 | SBR5 | SBR4 | SBR3 | SBR2 | SBR1 | SBR0 |
| | | W | | | | | | | | |
| 0x00BA | SCI2CR1[1] | R | LOOPS | SCISWAI | RSRC | M | WAKE | ILT | PE | PT |
| | | W | | | | | | | | |
| 0x00B8 | SCI2ASR1[(2)] | R | RXEDGIF | 0 | 0 | 0 | 0 | BERRV | BERRIF | BKDIF |
| | | W | | | | | | | | |
| 0x00B9 | SCI2ACR1[2] | R | RXEDGIE | 0 | 0 | 0 | 0 | 0 | BERRIE | BKDIE |
| | | W | | | | | | | | |
| 0x00BA | SCI2ACR2[2] | R | 0 | 0 | 0 | 0 | 0 | BERRM1 | BERRM0 | BKDFE |
| | | W | | | | | | | | |
| 0x00BB | SCI2CR2 | R | TIE | TCIE | RIE | ILIE | TE | RE | RWU | SBK |
| | | W | | | | | | | | |
| 0x00BC | SCI2SR1 | R | TDRE | TC | RDRF | IDLE | OR | NF | FE | PF |
| | | W | | | | | | | | |
| 0x00BD | SCI2SR2 | R | AMAP | 0 | 0 | TXPOL | RXPOL | BRK13 | TXDIR | RAF |
| | | W | | | | | | | | |
| 0x00BE | SCI2DRH | R | R8 | T8 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | W | | | | | | | | |
| 0x00BF | SCI2DRL | R | R7 | R6 | R5 | R4 | R3 | R2 | R1 | R0 |
| | | W | T7 | T6 | T5 | T4 | T3 | T2 | T1 | T0 |

1. Those registers are accessible if the AMAP bit in the SCI2SR2 register is set to zero
2. Those registers are accessible if the AMAP bit in the SCI2SR2 register is set to one

## 0x00C0–0x00C7 Asynchronous Serial Interface (SCI3) Map

| Address | Name | | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|---|---|
| 0x00C0 | SCI3BDH[1] | R | IREN | TNP1 | TNP0 | SBR12 | SBR11 | SBR10 | SBR9 | SBR8 |
| | | W | | | | | | | | |
| 0x00C1 | SCI3BDL[1] | R | SBR7 | SBR6 | SBR5 | SBR4 | SBR3 | SBR2 | SBR1 | SBR0 |
| | | W | | | | | | | | |
| 0x00C2 | SCI3CR1[1] | R | LOOPS | SCISWAI | RSRC | M | WAKE | ILT | PE | PT |
| | | W | | | | | | | | |
| 0x00C0 | SCI3ASR1[2] | R | RXEDGIF | 0 | 0 | 0 | 0 | BERRV | BERRIF | BKDIF |
| | | W | | | | | | | | |
| 0x00C1 | SCI3ACR1[2] | R | RXEDGIE | 0 | 0 | 0 | 0 | 0 | BERRIE | BKDIE |
| | | W | | | | | | | | |
| 0x00C2 | SCI3ACR2[2] | R | 0 | 0 | 0 | 0 | 0 | BERRM1 | BERRM0 | BKDFE |
| | | W | | | | | | | | |
| 0x00C3 | SCI3CR2 | R | TIE | TCIE | RIE | ILIE | TE | RE | RWU | SBK |
| | | W | | | | | | | | |
| 0x00C4 | SCI3SR1 | R | TDRE | TC | RDRF | IDLE | OR | NF | FE | PF |
| | | W | | | | | | | | |
| 0x00C5 | SCI3SR2 | R | AMAP | 0 | 0 | TXPOL | RXPOL | BRK13 | TXDIR | RAF |
| | | W | | | | | | | | |
| 0x00C6 | SCI3DRH | R | R8 | T8 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | W | | | | | | | | |
| 0x00C7 | SCI3DRL | R | R7 | R6 | R5 | R4 | R3 | R2 | R1 | R0 |
| | | W | T7 | T6 | T5 | T4 | T3 | T2 | T1 | T0 |

1. Those registers are accessible if the AMAP bit in the SCI3SR2 register is set to zero
2. Those registers are accessible if the AMAP bit in the SCI3SR2 register is set to one

## 0x00C8–0x00CF Asynchronous Serial Interface (SCI0) Map

| Address | Name | | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|---|---|
| 0x00C8 | SCI0BDH[1] | R | IREN | TNP1 | TNP0 | SBR12 | SBR11 | SBR10 | SBR9 | SBR8 |
| | | W | | | | | | | | |
| 0x00C9 | SCI0BDL[1] | R | SBR7 | SBR6 | SBR5 | SBR4 | SBR3 | SBR2 | SBR1 | SBR0 |
| | | W | | | | | | | | |
| 0x00CA | SCI0CR1[1] | R | LOOPS | SCISWAI | RSRC | M | WAKE | ILT | PE | PT |
| | | W | | | | | | | | |
| 0x00C8 | SCI0ASR1[2] | R | RXEDGIF | 0 | 0 | 0 | 0 | BERRV | BERRIF | BKDIF |
| | | W | | | | | | | | |
| 0x00C9 | SCI0ACR1[2] | R | RXEDGIE | 0 | 0 | 0 | 0 | 0 | BERRIE | BKDIE |
| | | W | | | | | | | | |
| 0x00CA | SCI0ACR2[2] | R | 0 | 0 | 0 | 0 | 0 | BERRM1 | BERRM0 | BKDFE |
| | | W | | | | | | | | |
| 0x00CB | SCI0CR2 | R | TIE | TCIE | RIE | ILIE | TE | RE | RWU | SBK |
| | | W | | | | | | | | |
| 0x00CC | SCI0SR1 | R | TDRE | TC | RDRF | IDLE | OR | NF | FE | PF |
| | | W | | | | | | | | |
| 0x00CD | SCI0SR2 | R | AMAP | 0 | 0 | TXPOL | RXPOL | BRK13 | TXDIR | RAF |
| | | W | | | | | | | | |
| 0x00CE | SCI0DRH | R | R8 | T8 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | W | | | | | | | | |
| 0x00CF | SCI0DRL | R | R7 | R6 | R5 | R4 | R3 | R2 | R1 | R0 |
| | | W | T7 | T6 | T5 | T4 | T3 | T2 | T1 | T0 |

1. Those registers are accessible if the AMAP bit in the SCI0SR2 register is set to zero
2. Those registers are accessible if the AMAP bit in the SCI0SR2 register is set to one

## 0x00D0–0x00D7 Asynchronous Serial Interface (SCI1) Map

| Address | Name | | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|---|---|
| 0x00D0 | SCI1BDH[1] | R | IREN | TNP1 | TNP0 | SBR12 | SBR11 | SBR10 | SBR9 | SBR8 |
| | | W | | | | | | | | |
| 0x00D1 | SCI1BDL[1] | R | SBR7 | SBR6 | SBR5 | SBR4 | SBR3 | SBR2 | SBR1 | SBR0 |
| | | W | | | | | | | | |
| 0x00D2 | SCI1CR1[1] | R | LOOPS | SCISWAI | RSRC | M | WAKE | ILT | PE | PT |
| | | W | | | | | | | | |
| 0x00D0 | SCI1ASR1[2] | R | RXEDGIF | 0 | 0 | 0 | 0 | BERRV | BERRIF | BKDIF |
| | | W | | | | | | | | |
| 0x00D1 | SCI1ACR1[2] | R | RXEDGIE | 0 | 0 | 0 | 0 | 0 | BERRIE | BKDIE |
| | | W | | | | | | | | |
| 0x00D2 | SCI1ACR2[2] | R | 0 | 0 | 0 | 0 | 0 | BERRM1 | BERRM0 | BKDFE |
| | | W | | | | | | | | |
| 0x00D3 | SCI1CR2 | R | TIE | TCIE | RIE | ILIE | TE | RE | RWU | SBK |
| | | W | | | | | | | | |
| 0x00D4 | SCI1SR1 | R | TDRE | TC | RDRF | IDLE | OR | NF | FE | PF |
| | | W | | | | | | | | |

## 0x00D0–0x00D7 Asynchronous Serial Interface (SCI1) Map (continued)

| Address | Name | | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|---|---|
| 0x00D5 | SCI1SR2 | R | AMAP | 0 | 0 | TXPOL | RXPOL | BRK13 | TXDIR | RAF |
| | | W | | | | | | | | |
| 0x00D6 | SCI1DRH | R | R8 | T8 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | W | | | | | | | | |
| 0x00D7 | SCI1DRL | R | R7 | R6 | R5 | R4 | R3 | R2 | R1 | R0 |
| | | W | T7 | T6 | T5 | T4 | T3 | T2 | T1 | T0 |

1. Those registers are accessible if the AMAP bit in the SCI1SR2 register is set to zero
2. Those registers are accessible if the AMAP bit in the SCI1SR2 register is set to one

## 0x00D8–0x00DF Serial Peripheral Interface (SPI0) Map

| Address | Name | | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|---|---|
| 0x00D8 | SPI0CR1 | R | SPIE | SPE | SPTIE | MSTR | CPOL | CPHA | SSOE | LSBFE |
| | | W | | | | | | | | |
| 0x00D9 | SPI0CR2 | R | 0 | XFRW | 0 | MODFEN | BIDIROE | 0 | SPISWAI | SPC0 |
| | | W | | | | | | | | |
| 0x00DA | SPI0BR | R | 0 | SPPR2 | SPPR1 | SPPR0 | 0 | SPR2 | SPR1 | SPR0 |
| | | W | | | | | | | | |
| 0x00DB | SPI0SR | R | SPIF | 0 | SPTEF | MODF | 0 | 0 | 0 | 0 |
| | | W | | | | | | | | |
| 0x00DC | SPI0DRH | R | R15 | R14 | R13 | R12 | R11 | R10 | R9 | R8 |
| | | W | T15 | T14 | T13 | T12 | T11 | T10 | T9 | T8 |
| 0x00DD | SPI0DRL | R | R7 | R6 | R5 | R4 | R3 | R2 | R1 | R0 |
| | | W | T7 | T6 | T5 | T4 | T3 | T2 | T1 | T0 |
| 0x00DE | Reserved | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | W | | | | | | | | |
| 0x00DF | Reserved | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | W | | | | | | | | |

## 0x00E0–0x00E7 Inter IC Bus (IIC0) Map

| Address | Name | | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|---|---|
| 0x00E0 | IBAD | R | ADR7 | ADR6 | ADR5 | ADR4 | ADR3 | ADR2 | ADR1 | 0 |
| | | W | | | | | | | | |
| 0x00E1 | IBFD | R | IBC7 | IBC6 | IBC5 | IBC4 | IBC3 | IBC2 | IBC1 | IBC0 |
| | | W | | | | | | | | |
| 0x00E2 | IBCR | R | IBEN | IBIE | MS/$\overline{SL}$ | TX/$\overline{RX}$ | TXAK | 0 | 0 | IBSWAI |
| | | W | | | | | | RSTA | | |
| 0x00E3 | IBSR | R | TCF | IAAS | IBB | IBAL | 0 | SRW | IBIF | RXAK |
| | | W | | | | | | | | |
| 0x00E4 | IBDR | R | D7 | D6 | D5 | D4 | D3 | D2 | D1 | D 0 |
| | | W | | | | | | | | |

## 0x00E0–0x00E7 Inter IC Bus (IIC0) Map (continued)

| Address | Name | | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|---|---|
| 0x00E5 | IBCR2 | R | GCEN | ADTYPE | 0 | 0 | 0 | ADR10 | ADR9 | ADR8 |
| | | W | | | | | | | | |
| 0x00E6 | Reserved | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | W | | | | | | | | |
| 0x00E7 | Reserved | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | W | | | | | | | | |

## 0x00E8–0x00EF Reserved

| Address | Name | | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|---|---|
| 0x00E8 | Reserved | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | W | | | | | | | | |
| 0x00E9 | Reserved | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | W | | | | | | | | |
| 0x00EA | Reserved | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | W | | | | | | | | |
| 0x00EB | Reserved | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | W | | | | | | | | |
| 0x00EC | Reserved | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | W | | | | | | | | |
| 0x00ED | Reserved | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | W | | | | | | | | |
| 0x00EE | Reserved | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | W | | | | | | | | |
| 0x00EF | Reserved | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | W | | | | | | | | |

## 0x00F0–0x00F7 Serial Peripheral Interface (SPI1) Map

| Address | Name | | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|---|---|
| 0x00F0 | SPI1CR1 | R | SPIE | SPE | SPTIE | MSTR | CPOL | CPHA | SSOE | LSBFE |
| | | W | | | | | | | | |
| 0x00F1 | SPI1CR2 | R | 0 | XFRW | 0 | MODFEN | BIDIROE | 0 | SPISWAI | SPC0 |
| | | W | | | | | | | | |
| 0x00F2 | SPI1BR | R | 0 | SPPR2 | SPPR1 | SPPR0 | 0 | SPR2 | SPR1 | SPR0 |
| | | W | | | | | | | | |
| 0x00F3 | SPI1SR | R | SPIF | 0 | SPTEF | MODF | 0 | 0 | 0 | 0 |
| | | W | | | | | | | | |
| 0x00F4 | SPI1DRH | R | R15 | R14 | R13 | R12 | R11 | R10 | R9 | R8 |
| | | W | T15 | T14 | T13 | T12 | T11 | T10 | T9 | T8 |
| 0x00F5 | SPI1DRL | R | R7 | R6 | R5 | R4 | R3 | R2 | R1 | R0 |
| | | W | T7 | T6 | T5 | T4 | T3 | T2 | T1 | T0 |
| 0x00F6 | Reserved | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | W | | | | | | | | |
| 0x00F7 | Reserved | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | W | | | | | | | | |

## 0x00F8–0x00FF Serial Peripheral Interface (SPI2) Map

| Address | Name | | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|---|---|
| 0x00F8 | SPI2CR1 | R | SPIE | SPE | SPTIE | MSTR | CPOL | CPHA | SSOE | LSBFE |
| | | W | | | | | | | | |
| 0x00F9 | SPI2CR2 | R | 0 | XFRW | 0 | MODFEN | BIDIROE | 0 | SPISWAI | SPC0 |
| | | W | | | | | | | | |
| 0x00FA | SPI2BR | R | 0 | SPPR2 | SPPR1 | SPPR0 | 0 | SPR2 | SPR1 | SPR0 |
| | | W | | | | | | | | |
| 0x00FB | SPI2SR | R | SPIF | 0 | SPTEF | MODF | 0 | 0 | 0 | 0 |
| | | W | | | | | | | | |
| 0x00FC | SPI2DRH | R | R15 | R14 | R13 | R12 | R11 | R10 | R9 | R8 |
| | | W | T15 | T14 | T13 | T12 | T11 | T10 | T9 | T8 |
| 0x00FD | SPI2DRL | R | R7 | R6 | R5 | R4 | R3 | R2 | R1 | R0 |
| | | W | T7 | T6 | T5 | T4 | T3 | T2 | T1 | T0 |
| 0x00FE | Reserved | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | W | | | | | | | | |
| 0x00FF | Reserved | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | W | | | | | | | | |

## 0x0100–0x0113 NVM Control Register (FTM) Map

| Address | Name | | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|---|---|
| 0x0100 | FCLKDIV | R | FDIVLD | FDIV6 | FDIV5 | FDIV4 | FDIV3 | FDIV2 | FDIV1 | FDIV0 |
| | | W | | | | | | | | |
| 0x0101 | FSEC | R | KEYEN1 | KEYEN0 | RNV5 | RNV4 | RNV3 | RNV2 | SEC1 | SEC0 |
| | | W | | | | | | | | |
| 0x0102 | FCCOBIX | R | 0 | 0 | 0 | 0 | 0 | CCOBIX2 | CCOBIX1 | CCOBIX0 |
| | | W | | | | | | | | |
| 0x0103 | FECCRIX | R | 0 | 0 | 0 | 0 | 0 | ECCRIX2 | ECCRIX1 | ECCRIX0 |
| | | W | | | | | | | | |
| 0x0104 | FCNFG | R | CCIE | 0 | 0 | IGNSF | 0 | 0 | FDFD | FSFD |
| | | W | | | | | | | | |
| 0x0105 | FERCNFG | R | ERSERIE | PGMERIE | EACCEIE | EPVIOLIE | ERSVIE1 | ERSVIE0 | DFDIE | SFDIE |
| | | W | | | | | | | | |
| 0x0106 | FSTAT | R | CCIF | 0 | ACCERR | FPVIOL | MGBUSY | RSVD | MGSTAT1 | MGSTAT0 |
| | | W | | | | | | | | |
| 0x0107 | FERSTAT | R | ERSERIF | PGMERIF | 0 | EPVIOLIF | ERSVIF1 | ERSVIF0 | DFDIF | SFDIF |
| | | W | | | | | | | | |
| 0x0108 | FPROT | R | FPOPEN | RNV6 | FPHDIS | FPHS1 | FPHS0 | FPLDIS | FPLS1 | FPLS0 |
| | | W | | | | | | | | |
| 0x0109 | EPROT | R | EPOPEN | RNV6 | RNV5 | RNV4 | EPDIS | EPS2 | EPS1 | EPS0 |
| | | W | | | | | | | | |
| 0x010A | FCCOBHI | R | CCOB15 | CCOB14 | CCOB13 | CCOB12 | CCOB11 | CCOB10 | CCOB9 | CCOB8 |
| | | W | | | | | | | | |
| 0x010B | FCCOBLO | R | CCOB7 | CCOB6 | CCOB5 | CCOB4 | CCOB3 | CCOB2 | CCOB1 | CCOB0 |
| | | W | | | | | | | | |

## 0x0100–0x0113 NVM Control Register (FTM) Map (continued)

| Address | Name | | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|---|---|
| 0x010C | ETAGHI | R | ETAG15 | ETAG14 | ETAG13 | ETAG12 | ETAG11 | ETAG10 | ETAG9 | ETAG8 |
| | | W | | | | | | | | |
| 0x010D | ETAGLO | R | ETAG7 | ETAG6 | ETAG5 | ETAG4 | ETAG3 | ETAG2 | ETAG1 | ETAG0 |
| | | W | | | | | | | | |
| 0x010E | FECCRHI | R | ECCR15 | ECCR14 | ECCR13 | ECCR12 | ECCR11 | ECCR10 | ECCR9 | ECCR8 |
| | | W | | | | | | | | |
| 0x010F | FECCRLO | R | ECCR7 | ECCR6 | ECCR5 | ECCR4 | ECCR3 | ECCR2 | ECCR1 | ECCR0 |
| | | W | | | | | | | | |
| 0x0110 | FOPT | R | NV7 | NV6 | NV5 | NV4 | NV3 | NV2 | NV1 | NV0 |
| | | W | | | | | | | | |
| 0x0111 | Reserved | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | W | | | | | | | | |
| 0x0112 | Reserved | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | W | | | | | | | | |
| 0x0113 | Reserved | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | W | | | | | | | | |

## 0x0114–0x011F Memory Protection Unit (MPU) Map

| Address | Name | | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|---|---|
| 0x0114 | MPUFLG | R | AEF | WPF | NEXF | 0 | 0 | 0 | 0 | SVSF |
| | | W | | | | | | | | |
| 0x0115 | MPUASTAT0 | R | 0 | ADDR[22:16] | | | | | | |
| | | W | | | | | | | | |
| 0x0116 | MPUASTAT1 | R | ADDR[15:8] | | | | | | | |
| | | W | | | | | | | | |
| 0x0117 | MPUASTAT2 | R | ADDR[7:0] | | | | | | | |
| | | W | | | | | | | | |
| 0x0118 | Reserved | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | W | | | | | | | | |
| 0x0119 | MPUSEL | R | SVSEN | 0 | 0 | 0 | 0 | SEL[2:0] | | |
| | | W | | | | | | | | |
| 0x011A | MPUDESC0[1] | R | MSTR0 | MSTR1 | MSTR2 | MSTR3 | LOW_ADDR[22:19] | | | |
| | | W | | | | | | | | |
| 0x011B | MPUDESC1[1] | R | LOW_ADDR[18:11] | | | | | | | |
| | | W | | | | | | | | |
| 0x011C | MPUDESC2[1] | R | LOW_ADDR[10:3] | | | | | | | |
| | | W | | | | | | | | |
| 0x011D | MPUDESC3[1] | R | WP | NEX | 0 | 0 | HIGH_ADDR[22:19] | | | |
| | | W | | | | | | | | |
| 0x011E | MPUDESC4[1] | R | HIGH_ADDR[18:11] | | | | | | | |
| | | W | | | | | | | | |
| 0x011F | MPUDESC5[1] | R | HIGH_ADDR[10:3] | | | | | | | |
| | | W | | | | | | | | |

1. The module addresses 0x03C6–0x03CB represent a window in the register map through which different descriptor registers are visible.

## 0x0120–0x012F Interrupt Module (S12XINT) Map

| Address | Name | | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|---|---|
| 0x0120 | Reserved | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | W | | | | | | | | |
| 0x0121 | IVBR | R | IVB_ADDR[7:0] | | | | | | | |
| | | W | | | | | | | | |
| 0x0122 | Reserved | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | W | | | | | | | | |
| 0x0123 | Reserved | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | W | | | | | | | | |
| 0x0124 | Reserved | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | W | | | | | | | | |
| 0x0125 | Reserved | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | W | | | | | | | | |
| 0x0126 | INT_XGPRIO | R | 0 | 0 | 0 | 0 | 0 | XILVL[2:0] | | |
| | | W | | | | | | | | |
| 0x0127 | INT_CFADDR | R | INT_CFADDR[7:4] | | | | 0 | 0 | 0 | 0 |
| | | W | | | | | | | | |

## 0x0120–0x012F Interrupt Module (S12XINT) Map (continued)

| Address | Name | | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|---|---|
| 0x0128 | INT_CFDATA0 | R | RQST | 0 | 0 | 0 | 0 | PRIOLVL[2:0] | | |
| | | W | | | | | | | | |
| 0x0129 | INT_CFDATA1 | R | RQST | 0 | 0 | 0 | 0 | PRIOLVL[2:0] | | |
| | | W | | | | | | | | |
| 0x012A | INT_CFDATA2 | R | RQST | 0 | 0 | 0 | 0 | PRIOLVL[2:0] | | |
| | | W | | | | | | | | |
| 0x012B | INT_CFDATA3 | R | RQST | 0 | 0 | 0 | 0 | PRIOLVL[2:0] | | |
| | | W | | | | | | | | |
| 0x012C | INT_CFDATA4 | R | RQST | 0 | 0 | 0 | 0 | PRIOLVL[2:0] | | |
| | | W | | | | | | | | |
| 0x012D | INT_CFDATA5 | R | RQST | 0 | 0 | 0 | 0 | PRIOLVL[2:0] | | |
| | | W | | | | | | | | |
| 0x012E | INT_CFDATA6 | R | RQST | 0 | 0 | 0 | 0 | PRIOLVL[2:0] | | |
| | | W | | | | | | | | |
| 0x012F | INT_CFDATA7 | R | RQST | 0 | 0 | 0 | 0 | PRIOLVL[2:0] | | |
| | | W | | | | | | | | |

## 0x00130–0x0137 Asynchronous Serial Interface (SCI4) Map

| Address | Name | | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|---|---|
| 0x0130 | SCI4BDH[1] | R | IREN | TNP1 | TNP0 | SBR12 | SBR11 | SBR10 | SBR9 | SBR8 |
| | | W | | | | | | | | |
| 0x0131 | SCI4BDL[1] | R | SBR7 | SBR6 | SBR5 | SBR4 | SBR3 | SBR2 | SBR1 | SBR0 |
| | | W | | | | | | | | |
| 0x0132 | SCI4CR1[1] | R | LOOPS | SCISWAI | RSRC | M | WAKE | ILT | PE | PT |
| | | W | | | | | | | | |
| 0x0130 | SCI4ASR1[2] | R | RXEDGIF | 0 | 0 | 0 | 0 | BERRV | BERRIF | BKDIF |
| | | W | | | | | | | | |
| 0x0131 | SCI4ACR1[2] | R | RXEDGIE | 0 | 0 | 0 | 0 | 0 | BERRIE | BKDIE |
| | | W | | | | | | | | |
| 0x0132 | SCI4ACR2[2] | R | 0 | 0 | 0 | 0 | 0 | BERRM1 | BERRM0 | BKDFE |
| | | W | | | | | | | | |
| 0x0133 | SCI4CR2 | R | TIE | TCIE | RIE | ILIE | TE | RE | RWU | SBK |
| | | W | | | | | | | | |
| 0x0134 | SCI4SR1 | R | TDRE | TC | RDRF | IDLE | OR | NF | FE | PF |
| | | W | | | | | | | | |
| 0x0135 | SCI4SR2 | R | AMAP | 0 | 0 | TXPOL | RXPOL | BRK13 | TXDIR | RAF |
| | | W | | | | | | | | |
| 0x0136 | SCI4DRH | R | R8 | T8 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | W | | | | | | | | |
| 0x0137 | SCI4DRL | R | R7 | R6 | R5 | R4 | R3 | R2 | R1 | R0 |
| | | W | T7 | T6 | T5 | T4 | T3 | T2 | T1 | T0 |

1. Those registers are accessible if the AMAP bit in the SCI4SR2 register is set to zero
2. Those registers are accessible if the AMAP bit in the SCI4SR2 register is set to one

## 0x0138–0x013F Asynchronous Serial Interface (SCI5) Map

| Address | Name | | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|---|---|
| 0x0138 | SCI5BDH[1] | R | IREN | TNP1 | TNP0 | SBR12 | SBR11 | SBR10 | SBR9 | SBR8 |
| | | W | | | | | | | | |
| 0x0139 | SCI5BDL[1] | R | SBR7 | SBR6 | SBR5 | SBR4 | SBR3 | SBR2 | SBR1 | SBR0 |
| | | W | | | | | | | | |
| 0x013A | SCI5CR1[1] | R | LOOPS | SCISWAI | RSRC | M | WAKE | ILT | PE | PT |
| | | W | | | | | | | | |
| 0x0138 | SCI5ASR1[2] | R | RXEDGIF | 0 | 0 | 0 | 0 | BERRV | BERRIF | BKDIF |
| | | W | | | | | | | | |
| 0x0139 | SCI5ACR1[2] | R | RXEDGIE | 0 | 0 | 0 | 0 | 0 | BERRIE | BKDIE |
| | | W | | | | | | | | |
| 0x013A | SCI5ACR2[2] | R | 0 | 0 | 0 | 0 | 0 | BERRM1 | BERRM0 | BKDFE |
| | | W | | | | | | | | |
| 0x013B | SCI5CR2 | R | TIE | TCIE | RIE | ILIE | TE | RE | RWU | SBK |
| | | W | | | | | | | | |
| 0x013C | SCI5SR1 | R | TDRE | TC | RDRF | IDLE | OR | NF | FE | PF |
| | | W | | | | | | | | |

## 0x0138–0x013F Asynchronous Serial Interface (SCI5) Map (continued)

| Address | Name | | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|---|---|
| 0x013D | SCI5SR2 | R | AMAP | 0 | 0 | TXPOL | RXPOL | BRK13 | TXDIR | RAF |
| | | W | | | | | | | | |
| 0x013E | SCI5DRH | R | R8 | T8 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | W | | | | | | | | |
| 0x013F | SCI5DRL | R | R7 | R6 | R5 | R4 | R3 | R2 | R1 | R0 |
| | | W | T7 | T6 | T5 | T4 | T3 | T2 | T1 | T0 |

1. Those registers are accessible if the AMAP bit in the SCI5SR2 register is set to zero
2. Those registers are accessible if the AMAP bit in the SCI5SR2 register is set to one

## 0x0140–0x017F MSCAN (CAN0) Map

| Address | Name | | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|---|---|
| 0x0140 | CAN0CTL0 | R | RXFRM | RXACT | CSWAI | SYNCH | TIME | WUPE | SLPRQ | INITRQ |
| | | W | | | | | | | | |
| 0x0141 | CAN0CTL1 | R | CANE | CLKSRC | LOOPB | LISTEN | BORM | WUPM | SLPAK | INITAK |
| | | W | | | | | | | | |
| 0x0142 | CAN0BTR0 | R | SJW1 | SJW0 | BRP5 | BRP4 | BRP3 | BRP2 | BRP1 | BRP0 |
| | | W | | | | | | | | |
| 0x0143 | CAN0BTR1 | R | SAMP | TSEG22 | TSEG21 | TSEG20 | TSEG13 | TSEG12 | TSEG11 | TSEG10 |
| | | W | | | | | | | | |
| 0x0144 | CAN0RFLG | R | WUPIF | CSCIF | RSTAT1 | RSTAT0 | TSTAT1 | TSTAT0 | OVRIF | RXF |
| | | W | | | | | | | | |
| 0x0145 | CAN0RIER | R | WUPIE | CSCIE | RSTATE1 | RSTATE0 | TSTATE1 | TSTATE0 | OVRIE | RXFIE |
| | | W | | | | | | | | |
| 0x0146 | CAN0TFLG | R | 0 | 0 | 0 | 0 | 0 | TXE2 | TXE1 | TXE0 |
| | | W | | | | | | | | |
| 0x0147 | CAN0TIER | R | 0 | 0 | 0 | 0 | 0 | TXEIE2 | TXEIE1 | TXEIE0 |
| | | W | | | | | | | | |
| 0x0148 | CAN0TARQ | R | 0 | 0 | 0 | 0 | 0 | ABTRQ2 | ABTRQ1 | ABTRQ0 |
| | | W | | | | | | | | |
| 0x0149 | CAN0TAAK | R | 0 | 0 | 0 | 0 | 0 | ABTAK2 | ABTAK1 | ABTAK0 |
| | | W | | | | | | | | |
| 0x014A | CAN0TBSEL | R | 0 | 0 | 0 | 0 | 0 | TX2 | TX1 | TX0 |
| | | W | | | | | | | | |
| 0x014B | CAN0IDAC | R | 0 | 0 | IDAM1 | IDAM0 | 0 | IDHIT2 | IDHIT1 | IDHIT0 |
| | | W | | | | | | | | |
| 0x014C | Reserved | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | W | | | | | | | | |
| 0x014D | CAN0MISC | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | BOHOLD |
| | | W | | | | | | | | |
| 0x014E | CAN0RXERR | R | RXERR7 | RXERR6 | RXERR5 | RXERR4 | RXERR3 | RXERR2 | RXERR1 | RXERR0 |
| | | W | | | | | | | | |
| 0x014F | CAN0TXERR | R | TXERR7 | TXERR6 | TXERR5 | TXERR4 | TXERR3 | TXERR2 | TXERR1 | TXERR0 |
| | | W | | | | | | | | |
| 0x0150–0x0153 | CAN0IDAR0–CAN0IDAR3 | R | AC7 | AC6 | AC5 | AC4 | AC3 | AC2 | AC1 | AC0 |
| | | W | | | | | | | | |

## 0x0140–0x017F MSCAN (CAN0) Map (continued)

| Address | Name | | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|---|---|
| 0x0154–0x0157 | CAN0IDMR0–CAN0IDMR3 | R | AM7 | AM6 | AM5 | AM4 | AM3 | AM2 | AM1 | AM0 |
| | | W | | | | | | | | |
| 0x0158–0x015B | CAN0IDAR4–CAN0IDAR7 | R | AC7 | AC6 | AC5 | AC4 | AC3 | AC2 | AC1 | AC0 |
| | | W | | | | | | | | |
| 0x015C–0x015F | CAN0IDMR4–CAN0IDMR7 | R | AM7 | AM6 | AM5 | AM4 | AM3 | AM2 | AM1 | AM0 |
| | | W | | | | | | | | |
| 0x0160–0x016F | CAN0RXFG | R | FOREGROUND RECEIVE BUFFER (See Detailed MSCAN Foreground Receive and Transmit Buffer Layout) | | | | | | | |
| | | W | | | | | | | | |
| 0x0170–0x017F | CAN0TXFG | R | FOREGROUND TRANSMIT BUFFER (See Detailed MSCAN Foreground Receive and Transmit Buffer Layout) | | | | | | | |
| | | W | | | | | | | | |

## Detailed MSCAN Foreground Receive and Transmit Buffer Layout

| Address | Name | | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|---|---|
| | Extended ID | R | ID28 | ID27 | ID26 | ID25 | ID24 | ID23 | ID22 | ID21 |
| 0xXXX0 | Standard ID | R | ID10 | ID9 | ID8 | ID7 | ID6 | ID5 | ID4 | ID3 |
| | CANxRIDR0 | W | | | | | | | | |
| | Extended ID | R | ID20 | ID19 | ID18 | SRR=1 | IDE=1 | ID17 | ID16 | ID15 |
| 0xXXX1 | Standard ID | R | ID2 | ID1 | ID0 | RTR | IDE=0 | | | |
| | CANxRIDR1 | W | | | | | | | | |
| | Extended ID | R | ID14 | ID13 | ID12 | ID11 | ID10 | ID9 | ID8 | ID7 |
| 0xXXX2 | Standard ID | R | | | | | | | | |
| | CANxRIDR2 | W | | | | | | | | |
| | Extended ID | R | ID6 | ID5 | ID4 | ID3 | ID2 | ID1 | ID0 | RTR |
| 0xXXX3 | Standard ID | R | | | | | | | | |
| | CANxRIDR3 | W | | | | | | | | |
| 0xXXX4–0xXXXB | CANxRDSR0–CANxRDSR7 | R | DB7 | DB6 | DB5 | DB4 | DB3 | DB2 | DB1 | DB0 |
| | | W | | | | | | | | |
| 0xXXXC | CANRxDLR | R | | | | | DLC3 | DLC2 | DLC1 | DLC0 |
| | | W | | | | | | | | |
| 0xXXXD | Reserved | R | | | | | | | | |
| | | W | | | | | | | | |
| 0xXXXE | CANxRTSRH | R | TSR15 | TSR14 | TSR13 | TSR12 | TSR11 | TSR10 | TSR9 | TSR8 |
| | | W | | | | | | | | |
| 0xXXXF | CANxRTSRL | R | TSR7 | TSR6 | TSR5 | TSR4 | TSR3 | TSR2 | TSR1 | TSR0 |
| | | W | | | | | | | | |
| 0xXX10 | Extended ID CANxTIDR0 | R | ID28 | ID27 | ID26 | ID25 | ID24 | ID23 | ID22 | ID21 |
| | | W | | | | | | | | |
| | Standard ID | R | ID10 | ID9 | ID8 | ID7 | ID6 | ID5 | ID4 | ID3 |
| | | W | | | | | | | | |

## Detailed MSCAN Foreground Receive and Transmit Buffer Layout (continued)

| Address | Name | | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|---|---|
| 0xXX0x XX10 | Extended ID CANxTIDR1 | R | ID20 | ID19 | ID18 | SRR=1 | IDE=1 | ID17 | ID16 | ID15 |
| | | W | | | | | | | | |
| | Standard ID | R | ID2 | ID1 | ID0 | RTR | IDE=0 | | | |
| | | W | | | | | | | | |
| 0xXX12 | Extended ID CANxTIDR2 | R | ID14 | ID13 | ID12 | ID11 | ID10 | ID9 | ID8 | ID7 |
| | | W | | | | | | | | |
| | Standard ID | R | | | | | | | | |
| | | W | | | | | | | | |
| 0xXX13 | Extended ID CANxTIDR3 | R | ID6 | ID5 | ID4 | ID3 | ID2 | ID1 | ID0 | RTR |
| | | W | | | | | | | | |
| | Standard ID | R | | | | | | | | |
| | | W | | | | | | | | |
| 0xXX14 – 0xXX1B | CANxTDSR0– CANxTDSR7 | R | DB7 | DB6 | DB5 | DB4 | DB3 | DB2 | DB1 | DB0 |
| | | W | | | | | | | | |
| 0xXX1C | CANxTDLR | R | | | | | DLC3 | DLC2 | DLC1 | DLC0 |
| | | W | | | | | | | | |
| 0xXX1D | CANxTTBPR | R | PRIO7 | PRIO6 | PRIO5 | PRIO4 | PRIO3 | PRIO2 | PRIO1 | PRIO0 |
| | | W | | | | | | | | |
| 0xXX1E | CANxTTSRH | R | TSR15 | TSR14 | TSR13 | TSR12 | TSR11 | TSR10 | TSR9 | TSR8 |
| | | W | | | | | | | | |
| 0xXX1F | CANxTTSRL | R | TSR7 | TSR6 | TSR5 | TSR4 | TSR3 | TSR2 | TSR1 | TSR0 |
| | | W | | | | | | | | |

## 0x0180–0x01BF MSCAN (CAN1) Map (Sheet 1 of 2)

| Address | Name | | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|---|---|
| 0x0180 | CAN1CTL0 | R | RXFRM | RXACT | CSWAI | SYNCH | TIME | WUPE | SLPRQ | INITRQ |
| | | W | | | | | | | | |
| 0x0181 | CAN1CTL1 | R | CANE | CLKSRC | LOOPB | LISTEN | BORM | WUPM | SLPAK | INITAK |
| | | W | | | | | | | | |
| 0x0182 | CAN1BTR0 | R | SJW1 | SJW0 | BRP5 | BRP4 | BRP3 | BRP2 | BRP1 | BRP0 |
| | | W | | | | | | | | |
| 0x0183 | CAN1BTR1 | R | SAMP | TSEG22 | TSEG21 | TSEG20 | TSEG13 | TSEG12 | TSEG11 | TSEG10 |
| | | W | | | | | | | | |
| 0x0184 | CAN1RFLG | R | WUPIF | CSCIF | RSTAT1 | RSTAT0 | TSTAT1 | TSTAT0 | OVRIF | RXF |
| | | W | | | | | | | | |
| 0x0185 | CAN1RIER | R | WUPIE | CSCIE | RSTATE1 | RSTATE0 | TSTATE1 | TSTATE0 | OVRIE | RXFIE |
| | | W | | | | | | | | |
| 0x0186 | CAN1TFLG | R | 0 | 0 | 0 | 0 | 0 | TXE2 | TXE1 | TXE0 |
| | | W | | | | | | | | |
| 0x0187 | CAN1TIER | R | 0 | 0 | 0 | 0 | 0 | TXEIE2 | TXEIE1 | TXEIE0 |
| | | W | | | | | | | | |
| 0x0188 | CAN1TARQ | R | 0 | 0 | 0 | 0 | 0 | ABTRQ2 | ABTRQ1 | ABTRQ0 |
| | | W | | | | | | | | |
| 0x0189 | CAN1TAAK | R | 0 | 0 | 0 | 0 | 0 | ABTAK2 | ABTAK1 | ABTAK0 |
| | | W | | | | | | | | |
| 0x018A | CAN1TBSEL | R | 0 | 0 | 0 | 0 | 0 | TX2 | TX1 | TX0 |
| | | W | | | | | | | | |
| 0x018B | CAN1IDAC | R | 0 | 0 | IDAM1 | IDAM0 | 0 | IDHIT2 | IDHIT1 | IDHIT0 |
| | | W | | | | | | | | |
| 0x018C | Reserved | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | W | | | | | | | | |
| 0x018D | CAN1MISC | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | BOHOLD |
| | | W | | | | | | | | |
| 0x018E | CAN1RXERR | R | RXERR7 | RXERR6 | RXERR5 | RXERR4 | RXERR3 | RXERR2 | RXERR1 | RXERR0 |
| | | W | | | | | | | | |
| 0x018F | CAN1TXERR | R | TXERR7 | TXERR6 | TXERR5 | TXERR4 | TXERR3 | TXERR2 | TXERR1 | TXERR0 |
| | | W | | | | | | | | |
| 0x0190 | CAN1IDAR0 | R | AC7 | AC6 | AC5 | AC4 | AC3 | AC2 | AC1 | AC0 |
| | | W | | | | | | | | |
| 0x0191 | CAN1IDAR1 | R | AC7 | AC6 | AC5 | AC4 | AC3 | AC2 | AC1 | AC0 |
| | | W | | | | | | | | |
| 0x0192 | CAN1IDAR2 | R | AC7 | AC6 | AC5 | AC4 | AC3 | AC2 | AC1 | AC0 |
| | | W | | | | | | | | |
| 0x0193 | CAN1IDAR3 | R | AC7 | AC6 | AC5 | AC4 | AC3 | AC2 | AC1 | AC0 |
| | | W | | | | | | | | |
| 0x0194 | CAN1IDMR0 | R | AM7 | AM6 | AM5 | AM4 | AM3 | AM2 | AM1 | AM0 |
| | | W | | | | | | | | |
| 0x0195 | CAN1IDMR1 | R | AM7 | AM6 | AM5 | AM4 | AM3 | AM2 | AM1 | AM0 |
| | | W | | | | | | | | |

## 0x0180–0x01BF MSCAN (CAN1) Map (Sheet 2 of 2)

| Address | Name | | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|---|---|
| 0x0196 | CAN1IDMR2 | R | AM7 | AM6 | AM5 | AM4 | AM3 | AM2 | AM1 | AM0 |
| | | W | | | | | | | | |
| 0x0197 | CAN1IDMR3 | R | AM7 | AM6 | AM5 | AM4 | AM3 | AM2 | AM1 | AM0 |
| | | W | | | | | | | | |
| 0x0198 | CAN1IDAR4 | R | AC7 | AC6 | AC5 | AC4 | AC3 | AC2 | AC1 | AC0 |
| | | W | | | | | | | | |
| 0x0199 | CAN1IDAR5 | R | AC7 | AC6 | AC5 | AC4 | AC3 | AC2 | AC1 | AC0 |
| | | W | | | | | | | | |
| 0x019A | CAN1IDAR6 | R | AC7 | AC6 | AC5 | AC4 | AC3 | AC2 | AC1 | AC0 |
| | | W | | | | | | | | |
| 0x019B | CAN1IDAR7 | R | AC7 | AC6 | AC5 | AC4 | AC3 | AC2 | AC1 | AC0 |
| | | W | | | | | | | | |
| 0x019C | CAN1IDMR4 | R | AM7 | AM6 | AM5 | AM4 | AM3 | AM2 | AM1 | AM0 |
| | | W | | | | | | | | |
| 0x019D | CAN1IDMR5 | R | AM7 | AM6 | AM5 | AM4 | AM3 | AM2 | AM1 | AM0 |
| | | W | | | | | | | | |
| 0x019E | CAN1IDMR6 | R | AM7 | AM6 | AM5 | AM4 | AM3 | AM2 | AM1 | AM0 |
| | | W | | | | | | | | |
| 0x019F | CAN1IDMR7 | R | AM7 | AM6 | AM5 | AM4 | AM3 | AM2 | AM1 | AM0 |
| | | W | | | | | | | | |
| 0x01A0–0x01AF | CAN1RXFG | R | FOREGROUND RECEIVE BUFFER (See Detailed MSCAN Foreground Receive and Transmit Buffer Layout) | | | | | | | |
| | | W | | | | | | | | |
| 0x01B0–0x01BF | CAN1TXFG | R | FOREGROUND TRANSMIT BUFFER (See Detailed MSCAN Foreground Receive and Transmit Buffer Layout) | | | | | | | |
| | | W | | | | | | | | |

## 0x01C0–0x01FF MSCAN (CAN2) Map (Sheet 1 of 3)

| Address | Name | | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|---|---|
| 0x01C0 | CAN2CTL0 | R | RXFRM | RXACT | CSWAI | SYNCH | TIME | WUPE | SLPRQ | INITRQ |
| | | W | | | | | | | | |
| 0x01C1 | CAN2CTL1 | R | CANE | CLKSRC | LOOPB | LISTEN | BORM | WUPM | SLPAK | INITAK |
| | | W | | | | | | | | |
| 0x01C2 | CAN2BTR0 | R | SJW1 | SJW0 | BRP5 | BRP4 | BRP3 | BRP2 | BRP1 | BRP0 |
| | | W | | | | | | | | |
| 0x01C3 | CAN2BTR1 | R | SAMP | TSEG22 | TSEG21 | TSEG20 | TSEG13 | TSEG12 | TSEG11 | TSEG10 |
| | | W | | | | | | | | |
| 0x01C4 | CAN2RFLG | R | WUPIF | CSCIF | RSTAT1 | RSTAT0 | TSTAT1 | TSTAT0 | OVRIF | RXF |
| | | W | | | | | | | | |
| 0x01C5 | CAN2RIER | R | WUPIE | CSCIE | RSTATE1 | RSTATE0 | TSTATE1 | TSTATE0 | OVRIE | RXFIE |
| | | W | | | | | | | | |
| 0x01C6 | CAN2TFLG | R | 0 | 0 | 0 | 0 | 0 | TXE2 | TXE1 | TXE0 |
| | | W | | | | | | | | |
| 0x01C7 | CAN2TIER | R | 0 | 0 | 0 | 0 | 0 | TXEIE2 | TXEIE1 | TXEIE0 |
| | | W | | | | | | | | |

## 0x01C0–0x01FF MSCAN (CAN2) Map (Sheet 2 of 3)

| Address | Name | | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|---|---|
| 0x01C8 | CAN2TARQ | R | 0 | 0 | 0 | 0 | 0 | ABTRQ2 | ABTRQ1 | ABTRQ0 |
| | | W | | | | | | | | |
| 0x01C9 | CAN2TAAK | R | 0 | 0 | 0 | 0 | 0 | ABTAK2 | ABTAK1 | ABTAK0 |
| | | W | | | | | | | | |
| 0x01CA | CAN2TBSEL | R | 0 | 0 | 0 | 0 | 0 | TX2 | TX1 | TX0 |
| | | W | | | | | | | | |
| 0x01CB | CAN2IDAC | R | 0 | 0 | IDAM1 | IDAM0 | 0 | IDHIT2 | IDHIT1 | IDHIT0 |
| | | W | | | | | | | | |
| 0x01CC | Reserved | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | W | | | | | | | | |
| 0x01CD | CAN2MISC | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | BOHOLD |
| | | W | | | | | | | | |
| 0x01CE | CAN2RXERR | R | RXERR7 | RXERR6 | RXERR5 | RXERR4 | RXERR3 | RXERR2 | RXERR1 | RXERR0 |
| | | W | | | | | | | | |
| 0x01CF | CAN2TXERR | R | TXERR7 | TXERR6 | TXERR5 | TXERR4 | TXERR3 | TXERR2 | TXERR1 | TXERR0 |
| | | W | | | | | | | | |
| 0x01D0 | CAN2IDAR0 | R | AC7 | AC6 | AC5 | AC4 | AC3 | AC2 | AC1 | AC0 |
| | | W | | | | | | | | |
| 0x01D1 | CAN2IDAR1 | R | AC7 | AC6 | AC5 | AC4 | AC3 | AC2 | AC1 | AC0 |
| | | W | | | | | | | | |
| 0x01D2 | CAN2IDAR2 | R | AC7 | AC6 | AC5 | AC4 | AC3 | AC2 | AC1 | AC0 |
| | | W | | | | | | | | |
| 0x01D3 | CAN2IDAR3 | R | AC7 | AC6 | AC5 | AC4 | AC3 | AC2 | AC1 | AC0 |
| | | W | | | | | | | | |
| 0x01D4 | CAN2IDMR0 | R | AM7 | AM6 | AM5 | AM4 | AM3 | AM2 | AM1 | AM0 |
| | | W | | | | | | | | |
| 0x01D5 | CAN2IDMR1 | R | AM7 | AM6 | AM5 | AM4 | AM3 | AM2 | AM1 | AM0 |
| | | W | | | | | | | | |
| 0x01D6 | CAN2IDMR2 | R | AM7 | AM6 | AM5 | AM4 | AM3 | AM2 | AM1 | AM0 |
| | | W | | | | | | | | |
| 0x01D7 | CAN2IDMR3 | R | AM7 | AM6 | AM5 | AM4 | AM3 | AM2 | AM1 | AM0 |
| | | W | | | | | | | | |
| 0x01D8 | CAN2IDAR4 | R | AC7 | AC6 | AC5 | AC4 | AC3 | AC2 | AC1 | AC0 |
| | | W | | | | | | | | |
| 0x01D9 | CAN2IDAR5 | R | AC7 | AC6 | AC5 | AC4 | AC3 | AC2 | AC1 | AC0 |
| | | W | | | | | | | | |
| 0x01DA | CAN2IDAR6 | R | AC7 | AC6 | AC5 | AC4 | AC3 | AC2 | AC1 | AC0 |
| | | W | | | | | | | | |
| 0x01DB | CAN2IDAR7 | R | AC7 | AC6 | AC5 | AC4 | AC3 | AC2 | AC1 | AC0 |
| | | W | | | | | | | | |
| 0x01DC | CAN2IDMR4 | R | AM7 | AM6 | AM5 | AM4 | AM3 | AM2 | AM1 | AM0 |
| | | W | | | | | | | | |
| 0x01DD | CAN2IDMR5 | R | AM7 | AM6 | AM5 | AM4 | AM3 | AM2 | AM1 | AM0 |
| | | W | | | | | | | | |
| 0x01DE | CAN2IDMR6 | R | AM7 | AM6 | AM5 | AM4 | AM3 | AM2 | AM1 | AM0 |
| | | W | | | | | | | | |

## 0x01C0–0x01FF MSCAN (CAN2) Map (Sheet 3 of 3)

| Address | Name | | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|---|---|
| 0x01DF | CAN2IDMR7 | R | AM7 | AM6 | AM5 | AM4 | AM3 | AM2 | AM1 | AM0 |
| | | W | | | | | | | | |
| 0x01E0–0x01EF | CAN2RXFG | R | FOREGROUND RECEIVE BUFFER (See Detailed MSCAN Foreground Receive and Transmit Buffer Layout) | | | | | | | |
| | | W | | | | | | | | |
| 0x01F0–0x01FF | CAN2TXFG | R | FOREGROUND TRANSMIT BUFFER (See Detailed MSCAN Foreground Receive and Transmit Buffer Layout) | | | | | | | |
| | | W | | | | | | | | |

## 0x0200–0x023F MSCAN (CAN3)

| Address | Name | | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|---|---|
| 0x0200 | CAN3CTL0 | R | RXFRM | RXACT | CSWAI | SYNCH | TIME | WUPE | SLPRQ | INITRQ |
| | | W | | | | | | | | |
| 0x0201 | CAN3CTL1 | R | CANE | CLKSRC | LOOPB | LISTEN | BORM | WUPM | SLPAK | INITAK |
| | | W | | | | | | | | |
| 0x0202 | CAN3BTR0 | R | SJW1 | SJW0 | BRP5 | BRP4 | BRP3 | BRP2 | BRP1 | BRP0 |
| | | W | | | | | | | | |
| 0x0203 | CAN3BTR1 | R | SAMP | TSEG22 | TSEG21 | TSEG20 | TSEG13 | TSEG12 | TSEG11 | TSEG10 |
| | | W | | | | | | | | |
| 0x0204 | CAN3RFLG | R | WUPIF | CSCIF | RSTAT1 | RSTAT0 | TSTAT1 | TSTAT0 | OVRIF | RXF |
| | | W | | | | | | | | |
| 0x0205 | CAN3RIER | R | WUPIE | CSCIE | RSTATE1 | RSTATE0 | TSTATE1 | TSTATE0 | OVRIE | RXFIE |
| | | W | | | | | | | | |
| 0x0206 | CAN3TFLG | R | 0 | 0 | 0 | 0 | 0 | TXE2 | TXE1 | TXE0 |
| | | W | | | | | | | | |
| 0x0207 | CAN3TIER | R | 0 | 0 | 0 | 0 | 0 | TXEIE2 | TXEIE1 | TXEIE0 |
| | | W | | | | | | | | |
| 0x0208 | CAN3TARQ | R | 0 | 0 | 0 | 0 | 0 | ABTRQ2 | ABTRQ1 | ABTRQ0 |
| | | W | | | | | | | | |
| 0x0209 | CAN3TAAK | R | 0 | 0 | 0 | 0 | 0 | ABTAK2 | ABTAK1 | ABTAK0 |
| | | W | | | | | | | | |
| 0x020A | CAN3TBSEL | R | 0 | 0 | 0 | 0 | 0 | TX2 | TX1 | TX0 |
| | | W | | | | | | | | |
| 0x020B | CAN3IDAC | R | 0 | 0 | IDAM1 | IDAM0 | 0 | IDHIT2 | IDHIT1 | IDHIT0 |
| | | W | | | | | | | | |
| 0x020C | Reserved | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | W | | | | | | | | |
| 0x020D | CAN3MISC | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | BOHOLD |
| | | W | | | | | | | | |
| 0x020E | CAN3RXERR | R | RXERR7 | RXERR6 | RXERR5 | RXERR4 | RXERR3 | RXERR2 | RXERR1 | RXERR0 |
| | | W | | | | | | | | |
| 0x020F | CAN3TXERR | R | TXERR7 | TXERR6 | TXERR5 | TXERR4 | TXERR3 | TXERR2 | TXERR1 | TXERR0 |
| | | W | | | | | | | | |
| 0x0210 | CAN3IDAR0 | R | AC7 | AC6 | AC5 | AC4 | AC3 | AC2 | AC1 | AC0 |
| | | W | | | | | | | | |
| 0x0211 | CAN3IDAR1 | R | AC7 | AC6 | AC5 | AC4 | AC3 | AC2 | AC1 | AC0 |
| | | W | | | | | | | | |
| 0x0212 | CAN3IDAR2 | R | AC7 | AC6 | AC5 | AC4 | AC3 | AC2 | AC1 | AC0 |
| | | W | | | | | | | | |
| 0x0213 | CAN3IDAR3 | R | AC7 | AC6 | AC5 | AC4 | AC3 | AC2 | AC1 | AC0 |
| | | W | | | | | | | | |
| 0x0214 | CAN3IDMR0 | R | AM7 | AM6 | AM5 | AM4 | AM3 | AM2 | AM1 | AM0 |
| | | W | | | | | | | | |
| 0x0215 | CAN3IDMR1 | R | AM7 | AM6 | AM5 | AM4 | AM3 | AM2 | AM1 | AM0 |
| | | W | | | | | | | | |

## 0x0200–0x023F MSCAN (CAN3) (continued)

| Address | Name | | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|---|---|
| 0x0216 | CAN3IDMR2 | R | AM7 | AM6 | AM5 | AM4 | AM3 | AM2 | AM1 | AM0 |
| | | W | | | | | | | | |
| 0x0217 | CAN3IDMR3 | R | AM7 | AM6 | AM5 | AM4 | AM3 | AM2 | AM1 | AM0 |
| | | W | | | | | | | | |
| 0x0218 | CAN3IDAR4 | R | AC7 | AC6 | AC5 | AC4 | AC3 | AC2 | AC1 | AC0 |
| | | W | | | | | | | | |
| 0x0219 | CAN3IDAR5 | R | AC7 | AC6 | AC5 | AC4 | AC3 | AC2 | AC1 | AC0 |
| | | W | | | | | | | | |
| 0x021A | CAN3IDAR6 | R | AC7 | AC6 | AC5 | AC4 | AC3 | AC2 | AC1 | AC0 |
| | | W | | | | | | | | |
| 0x021B | CAN3IDAR7 | R | AC7 | AC6 | AC5 | AC4 | AC3 | AC2 | AC1 | AC0 |
| | | W | | | | | | | | |
| 0x021C | CAN3IDMR4 | R | AM7 | AM6 | AM5 | AM4 | AM3 | AM2 | AM1 | AM0 |
| | | W | | | | | | | | |
| 0x021D | CAN3IDMR5 | R | AM7 | AM6 | AM5 | AM4 | AM3 | AM2 | AM1 | AM0 |
| | | W | | | | | | | | |
| 0x021E | CAN3IDMR6 | R | AM7 | AM6 | AM5 | AM4 | AM3 | AM2 | AM1 | AM0 |
| | | W | | | | | | | | |
| 0x021F | CAN3IDMR7 | R | AM7 | AM6 | AM5 | AM4 | AM3 | AM2 | AM1 | AM0 |
| | | W | | | | | | | | |
| 0x0220–0x022F | CAN3RXFG | R | FOREGROUND RECEIVE BUFFER (See Detailed MSCAN Foreground Receive and Transmit Buffer Layout) | | | | | | | |
| | | W | | | | | | | | |
| 0x0230–0x023F | CAN3TXFG | R | FOREGROUND TRANSMIT BUFFER (See Detailed MSCAN Foreground Receive and Transmit Buffer Layout) | | | | | | | |
| | | W | | | | | | | | |

## 0x0240–0x027F Port Integration Module (PIM) Map 5 of 6

| Address | Name | | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|---|---|
| 0x0240 | PTT | R | PTT7 | PTT6 | PTT5 | PTT4 | PTT3 | PTT2 | PTT1 | PTT0 |
| | | W | | | | | | | | |
| 0x0241 | PTIT | R | PTIT7 | PTIT6 | PTIT5 | PTIT4 | PTIT3 | PTIT2 | PTIT1 | PTIT0 |
| | | W | | | | | | | | |
| 0x0242 | DDRT | R | DDRT7 | DDRT7 | DDRT5 | DDRT4 | DDRT3 | DDRT2 | DDRT1 | DDRT0 |
| | | W | | | | | | | | |
| 0x0243 | RDRT | R | RDRT7 | RDRT6 | RDRT5 | RDRT4 | RDRT3 | RDRT2 | RDRT1 | RDRT0 |
| | | W | | | | | | | | |
| 0x0244 | PERT | R | PERT7 | PERT6 | PERT5 | PERT4 | PERT3 | PERT2 | PERT1 | PERT0 |
| | | W | | | | | | | | |
| 0x0245 | PPST | R | PPST7 | PPST6 | PPST5 | PPST4 | PPST3 | PPST2 | PPST1 | PPST0 |
| | | W | | | | | | | | |
| 0x0246 | Reserved | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | W | | | | | | | | |
| 0x0247 | Reserved | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | W | | | | | | | | |

## 0x0240–0x027F Port Integration Module (PIM) Map 5 of 6 (continued)

| Address | Name | | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|---|---|
| 0x0248 | PTS | R<br>W | PTS7 | PTS6 | PTS5 | PTS4 | PTS3 | PTS2 | PTS1 | PTS0 |
| 0x0249 | PTIS | R | PTIS7 | PTIS6 | PTIS5 | PTIS4 | PTIS3 | PTIS2 | PTIS1 | PTIS0 |
| | | W | | | | | | | | |
| 0x024A | DDRS | R<br>W | DDRS7 | DDRS7 | DDRS5 | DDRS4 | DDRS3 | DDRS2 | DDRS1 | DDRS0 |
| 0x024B | RDRS | R<br>W | RDRS7 | RDRS6 | RDRS5 | RDRS4 | RDRS3 | RDRS2 | RDRS1 | RDRS0 |
| 0x024C | PERS | R<br>W | PERS7 | PERS6 | PERS5 | PERS4 | PERS3 | PERS2 | PERS1 | PERS0 |
| 0x024D | PPSS | R<br>W | PPSS7 | PPSS6 | PPSS5 | PPSS4 | PPSS3 | PPSS2 | PPSS1 | PPSS0 |
| 0x024E | WOMS | R<br>W | WOMS7 | WOMS6 | WOMS5 | WOMS4 | WOMS3 | WOMS2 | WOMS1 | WOMS0 |
| 0x024F | Reserved | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | W | | | | | | | | |
| 0x0250 | PTM | R<br>W | PTM7 | PTM6 | PTM5 | PTM4 | PTM3 | PTM2 | PTM1 | PTM0 |
| 0x0251 | PTIM | R | PTIM7 | PTIM6 | PTIM5 | PTIM4 | PTIM3 | PTIM2 | PTIM1 | PTIM0 |
| | | W | | | | | | | | |
| 0x0252 | DDRM | R<br>W | DDRM7 | DDRM7 | DDRM5 | DDRM4 | DDRM3 | DDRM2 | DDRM1 | DDRM0 |
| 0x0253 | RDRM | R<br>W | RDRM7 | RDRM6 | RDRM5 | RDRM4 | RDRM3 | RDRM2 | RDRM1 | RDRM0 |
| 0x0254 | PERM | R<br>W | PERM7 | PERM6 | PERM5 | PERM4 | PERM3 | PERM2 | PERM1 | PERM0 |
| 0x0255 | PPSM | R<br>W | PPSM7 | PPSM6 | PPSM5 | PPSM4 | PPSM3 | PPSM2 | PPSM1 | PPSM0 |
| 0x0256 | WOMM | R<br>W | WOMM7 | WOMM6 | WOMM5 | WOMM4 | WOMM3 | WOMM2 | WOMM1 | WOMM0 |
| 0x0257 | MODRR | R | 0 | MODRR6 | MODRR5 | MODRR4 | MODRR3 | MODRR2 | MODRR1 | MODRR0 |
| | | W | | | | | | | | |
| 0x0258 | PTP | R<br>W | PTP7 | PTP6 | PTP5 | PTP4 | PTP3 | PTP2 | PTP1 | PTP0 |
| 0x0259 | PTIP | R | PTIP7 | PTIP6 | PTIP5 | PTIP4 | PTIP3 | PTIP2 | PTIP1 | PTIP0 |
| | | W | | | | | | | | |
| 0x025A | DDRP | R<br>W | DDRP7 | DDRP7 | DDRP5 | DDRP4 | DDRP3 | DDRP2 | DDRP1 | DDRP0 |
| 0x025B | RDRP | R<br>W | RDRP7 | RDRP6 | RDRP5 | RDRP4 | RDRP3 | RDRP2 | RDRP1 | RDRP0 |
| 0x025C | PERP | R<br>W | PERP7 | PERP6 | PERP5 | PERP4 | PERP3 | PERP2 | PERP1 | PERP0 |
| 0x025D | PPSP | R<br>W | PPSP7 | PPSP6 | PPSP5 | PPSP4 | PPSP3 | PPSP2 | PPSP1 | PPSS0 |
| 0x025E | PIEP | R<br>W | PIEP7 | PIEP6 | PIEP5 | PIEP4 | PIEP3 | PIEP2 | PIEP1 | PIEP0 |
| 0x025F | PIFP | R<br>W | PIFP7 | PIFP6 | PIFP5 | PIFP4 | PIFP3 | PIFP2 | PIFP1 | PIFP0 |

## 0x0240–0x027F Port Integration Module (PIM) Map 5 of 6 (continued)

| Address | Name | | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|---|---|
| 0x0260 | PTH | R<br>W | PTH7 | PTH6 | PTH5 | PTH4 | PTH3 | PTH2 | PTH1 | PTH0 |
| 0x0261 | PTIH | R | PTIH7 | PTIH6 | PTIH5 | PTIH4 | PTIH3 | PTIH2 | PTIH1 | PTIH0 |
| | | W | | | | | | | | |
| 0x0262 | DDRH | R<br>W | DDRH7 | DDRH7 | DDRH5 | DDRH4 | DDRH3 | DDRH2 | DDRH1 | DDRH0 |
| 0x0263 | RDRH | R<br>W | RDRH7 | RDRH6 | RDRH5 | RDRH4 | RDRH3 | RDRH2 | RDRH1 | RDRH0 |
| 0x0264 | PERH | R<br>W | PERH7 | PERH6 | PERH5 | PERH4 | PERH3 | PERH2 | PERH1 | PERH0 |
| 0x0265 | PPSH | R<br>W | PPSH7 | PPSH6 | PPSH5 | PPSH4 | PPSH3 | PPSH2 | PPSH1 | PPSH0 |
| 0x0266 | PIEH | R<br>W | PIEH7 | PIEH6 | PIEH5 | PIEH4 | PIEH3 | PIEH2 | PIEH1 | PIEH0 |
| 0x0267 | PIFH | R<br>W | PIFH7 | PIFH6 | PIFH5 | PIFH4 | PIFH3 | PIFH2 | PIFH1 | PIFH0 |
| 0x0268 | PTJ | R<br>W | PTJ7 | PTJ6 | PTJ5 | PTJ4 | PTJ3 | PTJ2 | PTJ1 | PTJ0 |
| 0x0269 | PTIJ | R | PTIJ7 | PTIJ6 | PTIJ5 | PTIJ4 | PTIJ3 | PTIJ2 | PTIJ1 | PTIJ0 |
| | | W | | | | | | | | |
| 0x026A | DDRJ | R<br>W | DDRJ7 | DDRJ7 | DDRJ5 | DDRJ4 | DDRJ3 | DDRJ2 | DDRJ1 | DDRJ0 |
| 0x026B | RDRJ | R<br>W | RDRJ7 | RDRJ6 | RDRJ5 | RDRJ4 | RDRJ3 | RDRJ2 | RDRJ1 | RDRJ0 |
| 0x026C | PERJ | R<br>W | PERJ7 | PERJ6 | PERJ5 | PERJ4 | PERJ3 | PERJ2 | PERJ1 | PERJ0 |
| 0x026D | PPSJ | R<br>W | PPSJ7 | PPSJ6 | PPSJ5 | PPSJ4 | PPSJ3 | PPSJ2 | PPSJ1 | PPSJ0 |
| 0x026E | PIEJ | R<br>W | PIEJ7 | PIEJ6 | PIEJ5 | PIEJ4 | PIEJ3 | PIEJ2 | PIEJ1 | PIEJ0 |
| 0x026f | PIFJ | R<br>W | PIFJ7 | PIFJ6 | PIFJ5 | PIFJ4 | PIFJ3 | PIFJ2 | PIFJ1 | PIFJ0 |
| 0x0270 | PT0AD0 | R<br>W | PT0AD07 | PT0AD06 | PT0AD05 | PT0AD04 | PT0AD03 | PT0AD02 | PT0AD01 | PT0AD00 |
| 0x0271 | PT1AD0 | R<br>W | PT1AD07 | PT1AD06 | PT1AD05 | PT1AD04 | PT1AD03 | PT1AD02 | PT1AD01 | PT1AD00 |
| 0x0272 | DDR0AD0 | R<br>W | DDR0AD07 | DDR0AD06 | DDR0AD05 | DDR0AD04 | DDR0AD03 | DDR0AD02 | DDR0AD01 | DDR0AD00 |
| 0x0273 | DDR1AD0 | R<br>W | DDR1AD07 | DDR1AD06 | DDR1AD05 | DDR1AD04 | DDR1AD03 | DDR1AD02 | DDR1AD01 | DDR1AD00 |
| 0x0274 | RDR0AD0 | R<br>W | RDR0AD07 | RDR0AD06 | RDR0AD05 | RDR0AD04 | RDR0AD03 | RDR0AD02 | RDR0AD01 | RDR0AD00 |
| 0x0275 | RDR1AD0 | R<br>W | RDR1AD07 | RDR1AD06 | RDR1AD05 | RDR1AD04 | RDR1AD03 | RDR1AD02 | RDR1AD01 | RDR1AD00 |
| 0x0276 | PER0AD0 | R<br>W | PER0AD07 | PER0AD06 | PER0AD05 | PER0AD04 | PER0AD03 | PER0AD02 | PER0AD01 | PER0AD00 |
| 0x0277 | PER1AD0 | R<br>W | PER1AD07 | PER1AD06 | PER1AD05 | PER1AD04 | PER1AD03 | PER1AD02 | PER1AD01 | PER1AD00 |

## 0x0240–0x027F Port Integration Module (PIM) Map 5 of 6 (continued)

| Address | Name | | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|---|---|
| 0x0278 | PT0AD1 | R<br>W | PT0AD17 | PT0AD16 | PT0AD15 | PT0AD14 | PT0AD13 | PT0AD12 | PT0AD11 | PT0AD10 |
| 0x0279 | PT1AD1 | R<br>W | PT1AD17 | PT1AD16 | PT1AD15 | PT1AD14 | PT1AD13 | PT1AD12 | PT1AD11 | PT1AD10 |
| 0x027A | DDR0AD1 | R<br>W | DDR0AD17 | DDR0AD16 | DDR0AD15 | DDR0AD14 | DDR0AD13 | DDR0AD12 | DDR0AD11 | DDR0AD10 |
| 0x027B | DDR1AD1 | R<br>W | DDR1AD17 | DDR1AD16 | DDR1AD15 | DDR1AD14 | DDR1AD13 | DDR1AD12 | DDR1AD11 | DDR1AD10 |
| 0x027C | RDR0AD1 | R<br>W | RDR0AD17 | RDR0AD16 | RDR0AD15 | RDR0AD14 | RDR0AD13 | RDR0AD12 | RDR0AD11 | RDR0AD10 |
| 0x027D | RDR1AD1 | R<br>W | RDR1AD17 | RDR1AD16 | RDR1AD15 | RDR1AD14 | RDR1AD13 | RDR1AD12 | RDR1AD11 | RDR1AD10 |
| 0x027E | PER0AD1 | R<br>W | PER0AD17 | PER0AD16 | PER0AD15 | PER0AD14 | PER0AD13 | PER0AD12 | PER0AD11 | PER0AD10 |
| 0x027F | PER1AD1 | R<br>W | PER1AD17 | PER1AD16 | PER1AD15 | PER1AD14 | PER1AD13 | PER1A1D2 | PER1AD11 | PER1AD10 |

## 0x0280–0x02BF MSCAN (CAN4) Map

| Address | Name | | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|---|---|
| 0x0280 | CAN4CTL0 | R<br>W | RXFRM | RXACT | CSWAI | SYNCH | TIME | WUPE | SLPRQ | INITRQ |
| 0x0281 | CAN4CTL1 | R<br>W | CANE | CLKSRC | LOOPB | LISTEN | BORM | WUPM | SLPAK | INITAK |
| 0x0282 | CAN4BTR0 | R<br>W | SJW1 | SJW0 | BRP5 | BRP4 | BRP3 | BRP2 | BRP1 | BRP0 |
| 0x0283 | CAN4BTR1 | R<br>W | SAMP | TSEG22 | TSEG21 | TSEG20 | TSEG13 | TSEG12 | TSEG11 | TSEG10 |
| 0x0284 | CAN4RFLG | R<br>W | WUPIF | CSCIF | RSTAT1 | RSTAT0 | TSTAT1 | TSTAT0 | OVRIF | RXF |
| 0x0285 | CAN4RIER | R<br>W | WUPIE | CSCIE | RSTATE1 | RSTATE0 | TSTATE1 | TSTATE0 | OVRIE | RXFIE |
| 0x0286 | CAN4TFLG | R<br>W | 0 | 0 | 0 | 0 | 0 | TXE2 | TXE1 | TXE0 |
| 0x0287 | CAN4TIER | R<br>W | 0 | 0 | 0 | 0 | 0 | TXEIE2 | TXEIE1 | TXEIE0 |
| 0x0288 | CAN4TARQ | R<br>W | 0 | 0 | 0 | 0 | 0 | ABTRQ2 | ABTRQ1 | ABTRQ0 |
| 0x0289 | CAN4TAAK | R<br>W | 0 | 0 | 0 | 0 | 0 | ABTAK2 | ABTAK1 | ABTAK0 |
| 0x028A | CAN4TBSEL | R<br>W | 0 | 0 | 0 | 0 | 0 | TX2 | TX1 | TX0 |
| 0x028B | CAN4IDAC | R<br>W | 0 | 0 | IDAM1 | IDAM0 | 0 | IDHIT2 | IDHIT1 | IDHIT0 |
| 0x028C | Reserved | R<br>W | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

## 0x0280–0x02BF MSCAN (CAN4) Map (continued)

| Address | Name | | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|---|---|
| 0x028D | CAN4MISC | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | BOHOLD |
| | | W | | | | | | | | |
| 0x028E | CAN4RXERR | R | RXERR7 | RXERR6 | RXERR5 | RXERR4 | RXERR3 | RXERR2 | RXERR1 | RXERR0 |
| | | W | | | | | | | | |
| 0x028F | CAN4TXERR | R | TXERR7 | TXERR6 | TXERR5 | TXERR4 | TXERR3 | TXERR2 | TXERR1 | TXERR0 |
| | | W | | | | | | | | |
| 0x0290 | CAN4IDAR0 | R | AC7 | AC6 | AC5 | AC4 | AC3 | AC2 | AC1 | AC0 |
| | | W | | | | | | | | |
| 0x0291 | CAN4IDAR1 | R | AC7 | AC6 | AC5 | AC4 | AC3 | AC2 | AC1 | AC0 |
| | | W | | | | | | | | |
| 0x0292 | CAN4IDAR2 | R | AC7 | AC6 | AC5 | AC4 | AC3 | AC2 | AC1 | AC0 |
| | | W | | | | | | | | |
| 0x0293 | CAN4IDAR3 | R | AC7 | AC6 | AC5 | AC4 | AC3 | AC2 | AC1 | AC0 |
| | | W | | | | | | | | |
| 0x0294 | CAN4IDMR0 | R | AM7 | AM6 | AM5 | AM4 | AM3 | AM2 | AM1 | AM0 |
| | | W | | | | | | | | |
| 0x0295 | CAN4IDMR1 | R | AM7 | AM6 | AM5 | AM4 | AM3 | AM2 | AM1 | AM0 |
| | | W | | | | | | | | |
| 0x0296 | CAN4IDMR2 | R | AM7 | AM6 | AM5 | AM4 | AM3 | AM2 | AM1 | AM0 |
| | | W | | | | | | | | |
| 0x0297 | CAN4IDMR3 | R | AM7 | AM6 | AM5 | AM4 | AM3 | AM2 | AM1 | AM0 |
| | | W | | | | | | | | |
| 0x0298 | CAN4IDAR4 | R | AC7 | AC6 | AC5 | AC4 | AC3 | AC2 | AC1 | AC0 |
| | | W | | | | | | | | |
| 0x0299 | CAN4IDAR5 | R | AC7 | AC6 | AC5 | AC4 | AC3 | AC2 | AC1 | AC0 |
| | | W | | | | | | | | |
| 0x029A | CAN4IDAR6 | R | AC7 | AC6 | AC5 | AC4 | AC3 | AC2 | AC1 | AC0 |
| | | W | | | | | | | | |
| 0x029B | CAN4IDAR7 | R | AC7 | AC6 | AC5 | AC4 | AC3 | AC2 | AC1 | AC0 |
| | | W | | | | | | | | |
| 0x029C | CAN4IDMR4 | R | AM7 | AM6 | AM5 | AM4 | AM3 | AM2 | AM1 | AM0 |
| | | W | | | | | | | | |
| 0x029D | CAN4IDMR5 | R | AM7 | AM6 | AM5 | AM4 | AM3 | AM2 | AM1 | AM0 |
| | | W | | | | | | | | |
| 0x029E | CAN4IDMR6 | R | AM7 | AM6 | AM5 | AM4 | AM3 | AM2 | AM1 | AM0 |
| | | W | | | | | | | | |
| 0x029F | CAN4IDMR7 | R | AM7 | AM6 | AM5 | AM4 | AM3 | AM2 | AM1 | AM0 |
| | | W | | | | | | | | |
| 0x02A0–0x02AF | CAN4RXFG | R | FOREGROUND RECEIVE BUFFER (See Detailed MSCAN Foreground Receive and Transmit Buffer Layout) | | | | | | | |
| | | W | | | | | | | | |
| 0x02B0–0x02BF | CAN4TXFG | R | FOREGROUND TRANSMIT BUFFER (See Detailed MSCAN Foreground Receive and Transmit Buffer Layout) | | | | | | | |
| | | W | | | | | | | | |

## 0x02C0–0x02EF Analog-to-Digital Converter 12-Bit 16-Channel (ATD0) Map

| Address | Name | | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|---|---|
| 0x02C0 | ATD0CTL0 | R | 0 | 0 | 0 | 0 | WRAP3 | WRAP2 | WRAP1 | WRAP0 |
| | | W | | | | | | | | |
| 0x02C1 | ATD0CTL1 | R | ETRIG SEL | SRES1 | SRES0 | SMP_DIS | ETRIG CH3 | ETRIG CH2 | ETRIG CH1 | ETRIG CH0 |
| | | W | | | | | | | | |
| 0x02C2 | ATD0CTL2 | R | 0 | AFFC | ICLKSTP | ETRIGLE | ETRIGP | ETRIGE | ASCIE | ACMPIE |
| | | W | | | | | | | | |
| 0x02C3 | ATD0CTL3 | R | DJM | S8C | S4C | S2C | S1C | FIFO | FRZ1 | FRZ0 |
| | | W | | | | | | | | |
| 0x02C4 | ATD0CTL4 | R | SMP2 | SMP1 | SMP0 | PRS4 | PRS3 | PRS2 | PRS1 | PRS0 |
| | | W | | | | | | | | |
| 0x02C5 | ATD0CTL5 | R | 0 | SC | SCAN | MULT | CD | CC | CB | CA |
| | | W | | | | | | | | |
| 0x02C6 | ATD0STAT0 | R | SCF | 0 | ETORF | FIFOR | CC3 | CC2 | CC1 | CC0 |
| | | W | | | | | | | | |
| 0x02C7 | Reserved | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | W | | | | | | | | |
| 0x02C8 | ATD0CMPEH | R | CMPE15 | CMPE14 | CMPE13 | CMPE12 | CMPE11 | CMPE10 | CMPE9 | CMPE8 |
| | | W | | | | | | | | |
| 0x02C9 | ATD0CMPEL | R | CMPE7 | CMPE6 | CMPE5 | CMPE4 | CMPE3 | CMPE2 | CMPE1 | CMPE0 |
| | | W | | | | | | | | |
| 0x02CA | ATD0STAT2H | R | CCF15 | CCF14 | CCF13 | CCF12 | CCF11 | CCF10 | CCF9 | CCF8 |
| | | W | | | | | | | | |
| 0x02CB | ATD0STAT2L | R | CCF7 | CCF6 | CCF5 | CCF4 | CCF3 | CCF2 | CCF1 | CCF0 |
| | | W | | | | | | | | |
| 0x02CC | ATD0DIENH | R | IEN15 | IEN14 | IEN13 | IEN12 | IEN11 | IEN10 | IEN9 | IEN8 |
| | | W | | | | | | | | |
| 0x02CD | ATD0DIENL | R | IEN7 | IEN6 | IEN5 | IEN4 | IEN3 | IEN2 | IEN1 | IEN0 |
| | | W | | | | | | | | |
| 0x02CE | ATD0CMPHTH | R | CMPHT15 | CMPHT14 | CMPHT13 | CMPHT12 | CMPHT11 | CMPHT10 | CMPHT9 | CMPHT8 |
| | | W | | | | | | | | |
| 0x02CF | ATD0CMPHTL | R | CMPHT7 | CMPHT6 | CMPHT5 | CMPHT4 | CMPHT3 | CMPHT2 | CMPHT1 | CMPHT0 |
| | | W | | | | | | | | |
| 0x02D0 | ATD0DR0H | R | Bit15 | 14 | 13 | 12 | 11 | 10 | 9 | Bit8 |
| | | W | | | | | | | | |
| 0x02D1 | ATD0DR0L | R | Bit7 | Bit6 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | W | | | | | | | | |
| 0x02D2 | ATD0DR1H | R | Bit15 | 14 | 13 | 12 | 11 | 10 | 9 | Bit8 |
| | | W | | | | | | | | |
| 0x02D3 | ATD0DR1L | R | Bit7 | Bit6 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | W | | | | | | | | |
| 0x02D4 | ATD0DR2H | R | Bit15 | 14 | 13 | 12 | 11 | 10 | 9 | Bit8 |
| | | W | | | | | | | | |
| 0x02D5 | ATD0DR2L | R | Bit7 | Bit6 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | W | | | | | | | | |

## 0x02C0–0x02EF Analog-to-Digital Converter 12-Bit 16-Channel (ATD0) Map (continued)

| Address | Name | | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|---|---|
| 0x02D6 | ATD0DR3H | R | Bit15 | 14 | 13 | 12 | 11 | 10 | 9 | Bit8 |
| | | W | | | | | | | | |
| 0x02D7 | ATD0DR3L | R | Bit7 | Bit6 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | W | | | | | | | | |
| 0x02D8 | ATD0DR4H | R | Bit15 | 14 | 13 | 12 | 11 | 10 | 9 | Bit8 |
| | | W | | | | | | | | |
| 0x02D9 | ATD0DR4L | R | Bit7 | Bit6 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | W | | | | | | | | |
| 0x02DA | ATD0DR5H | R | Bit15 | 14 | 13 | 12 | 11 | 10 | 9 | Bit8 |
| | | W | | | | | | | | |
| 0x02DB | ATD0DR5L | R | Bit7 | Bit6 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | W | | | | | | | | |
| 0x02DC | ATD0DR6H | R | Bit15 | 14 | 13 | 12 | 11 | 10 | 9 | Bit8 |
| | | W | | | | | | | | |
| 0x02DD | ATD0DR6L | R | Bit7 | Bit6 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | W | | | | | | | | |
| 0x02DE | ATD0DR7H | R | Bit15 | 14 | 13 | 12 | 11 | 10 | 9 | Bit8 |
| | | W | | | | | | | | |
| 0x02DF | ATD0DR7L | R | Bit7 | Bit6 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | W | | | | | | | | |
| 0x02E0 | ATD0DR8H | R | Bit15 | 14 | 13 | 12 | 11 | 10 | 9 | Bit8 |
| | | W | | | | | | | | |
| 0x02E1 | ATD0DR8L | R | Bit7 | Bit6 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | W | | | | | | | | |
| 0x02E2 | ATD0DR9H | R | Bit15 | 14 | 13 | 12 | 11 | 10 | 9 | Bit8 |
| | | W | | | | | | | | |
| 0x02E3 | ATD0DR9L | R | Bit7 | Bit6 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | W | | | | | | | | |
| 0x02E4 | ATD0DR10H | R | Bit15 | 14 | 13 | 12 | 11 | 10 | 9 | Bit8 |
| | | W | | | | | | | | |
| 0x02E5 | ATD0DR10L | R | Bit7 | Bit6 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | W | | | | | | | | |
| 0x02E6 | ATD0DR11H | R | Bit15 | 14 | 13 | 12 | 11 | 10 | 9 | Bit8 |
| | | W | | | | | | | | |
| 0x02E7 | ATD0DR11L | R | Bit7 | Bit6 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | W | | | | | | | | |
| 0x02E8 | ATD0DR12H | R | Bit15 | 14 | 13 | 12 | 11 | 10 | 9 | Bit8 |
| | | W | | | | | | | | |
| 0x02E9 | ATD0DR12L | R | Bit7 | Bit6 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | W | | | | | | | | |
| 0x02EA | ATD0DR13H | R | Bit15 | 14 | 13 | 12 | 11 | 10 | 9 | Bit8 |
| | | W | | | | | | | | |
| 0x02EB | ATD0DR13L | R | Bit7 | Bit6 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | W | | | | | | | | |
| 0x02EC | ATD0DR14H | R | Bit15 | 14 | 13 | 12 | 11 | 10 | 9 | Bit8 |
| | | W | | | | | | | | |

## 0x02C0–0x02EF Analog-to-Digital Converter 12-Bit 16-Channel (ATD0) Map (continued)

| Address | Name | | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|---|---|
| 0x02ED | ATD0DR14L | R | Bit7 | Bit6 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | W | | | | | | | | |
| 0x02EE | ATD0DR15H | R | Bit15 | 14 | 13 | 12 | 11 | 10 | 9 | Bit8 |
| | | W | | | | | | | | |
| 0x02EF | ATD0DR15L | R | Bit7 | Bit6 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | W | | | | | | | | |

## 0x02F0–0x02F7 Voltage Regulator (VREG_3V3) Map

| Address | Name | | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|---|---|
| 0x02F0 | VREGHTCL | R | 0 | 0 | VSEL | VAE | HTEN | HTDS | HTIE | HTIF |
| | | W | | | | | | | | |
| 0x02F1 | VREGCTRL | R | 0 | 0 | 0 | 0 | 0 | LVDS | LVIE | LVIF |
| | | W | | | | | | | | |
| 0x02F2 | VREGAPICL | R | APICLK | 0 | 0 | APIFES | APIEA | APIFE | APIE | APIF |
| | | W | | | | | | | | |
| 0x02F3 | VREGAPITR | R | APITR5 | APITR4 | APITR3 | APITR2 | APITR1 | APITR0 | 0 | 0 |
| | | W | | | | | | | | |
| 0x02F4 | VREGAPIRH | R | APIR15 | APIR14 | APIR13 | APIR12 | APIR11 | APIR10 | APIR9 | APIR8 |
| | | W | | | | | | | | |
| 0x02F5 | VREGAPIRL | R | APIR7 | APIR6 | APIR5 | APIR4 | APIR3 | APIR2 | APIR1 | APIR0 |
| | | W | | | | | | | | |
| 0x02F6 | Reserved | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | W | | | | | | | | |
| 0x02F7 | VREGHTTR | R | HTOEN | 0 | 0 | 0 | HTTR3 | HTTR2 | HTTR1 | HTTR0 |
| | | W | | | | | | | | |

## 0x02F8–0x02FF Reserved

| Address | Name | | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|---|---|
| 0x02F8–0x02FF | Reserved | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | W | | | | | | | | |

## 0x0300–0x0327 Pulse Width Modulator 8-Bit 8-Channel (PWM) Map (Sheet 1 of 3)

| Address | Name | | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|---|---|
| 0x0300 | PWME | R | PWME7 | PWME6 | PWME5 | PWME4 | PWME3 | PWME2 | PWME1 | PWME0 |
| | | W | | | | | | | | |
| 0x0301 | PWMPOL | R | PPOL7 | PPOL6 | PPOL5 | PPOL4 | PPOL3 | PPOL2 | PPOL1 | PPOL0 |
| | | W | | | | | | | | |
| 0x0302 | PWMCLK | R | PCLK7 | PCLK6 | PCLK5 | PCLK4 | PCLK3 | PCLK2 | PCLK1 | PCLK0 |
| | | W | | | | | | | | |
| 0x0303 | PWMPRCLK | R | 0 | PCKB2 | PCKB1 | PCKB0 | 0 | PCKA2 | PCKA1 | PCKA0 |
| | | W | | | | | | | | |
| 0x0304 | PWMCAE | R | CAE7 | CAE6 | CAE5 | CAE4 | CAE3 | CAE2 | CAE1 | CAE0 |
| | | W | | | | | | | | |
| 0x0305 | PWMCTL | R | CON67 | CON45 | CON23 | CON01 | PSWAI | PFRZ | 0 | 0 |
| | | W | | | | | | | | |
| 0x0306 | PWMTST Test Only | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | W | | | | | | | | |
| 0x0307 | PWMPRSC | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | W | | | | | | | | |
| 0x0308 | PWMSCLA | R | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
| | | W | | | | | | | | |
| 0x0309 | PWMSCLB | R | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
| | | W | | | | | | | | |

## 0x0300–0x0327 Pulse Width Modulator 8-Bit 8-Channel (PWM) Map (Sheet 2 of 3)

| Address | Name | | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|---|---|
| 0x030A | PWMSCNTA | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | W | | | | | | | | |
| 0x030B | PWMSCNTB | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | W | | | | | | | | |
| 0x030C | PWMCNT0 | R | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
| | | W | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0x030D | PWMCNT1 | R | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
| | | W | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0x030E | PWMCNT2 | R | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
| | | W | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0x030F | PWMCNT3 | R | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
| | | W | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0x0310 | PWMCNT4 | R | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
| | | W | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0x0311 | PWMCNT5 | R | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
| | | W | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0x0312 | PWMCNT6 | R | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
| | | W | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0x0313 | PWMCNT7 | R | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
| | | W | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0x0314 | PWMPER0 | R<br>W | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
| 0x0315 | PWMPER1 | R<br>W | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
| 0x0316 | PWMPER2 | R<br>W | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
| 0x0317 | PWMPER3 | R<br>W | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
| 0x0318 | PWMPER4 | R<br>W | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
| 0x0319 | PWMPER5 | R<br>W | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
| 0x031A | PWMPER6 | R<br>W | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
| 0x031B | PWMPER7 | R<br>W | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
| 0x031C | PWMDTY0 | R<br>W | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
| 0x031D | PWMDTY1 | R<br>W | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
| 0x031E | PWMDTY2 | R<br>W | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
| 0x031F | PWMDTY3 | R<br>W | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
| 0x0320 | PWMDTY4 | R<br>W | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |

## 0x0300–0x0327 Pulse Width Modulator 8-Bit 8-Channel (PWM) Map (Sheet 3 of 3)

| Address | Name | | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|---|---|
| 0x0321 | PWMDTY5 | R<br>W | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
| 0x0322 | PWMDTY6 | R<br>W | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
| 0x0323 | PWMDTY7 | R<br>W | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
| 0x0324 | PWMSDN | R<br>W | PWMIF | PWMIE | 0<br>PWM RSTRT | PWMLVL | 0 | PWM7IN | PWM7INL | PWM7 ENA |
| 0x0325 | Reserved | R<br>W | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0x0326 | Reserved | R<br>W | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0x0327 | Reserved | R<br>W | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

## 0x0328–0x032F Reserved

| Address | Name | | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|---|---|
| 0x0328–<br>0x032F | Reserved | R<br>W | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

## 0x00330–0x0337 Asynchronous Serial Interface (SCI6) Map

| Address | Name | | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|---|---|
| 0x0330 | SCI6BDH[(1)] | R<br>W | IREN | TNP1 | TNP0 | SBR12 | SBR11 | SBR10 | SBR9 | SBR8 |
| 0x0331 | SCI6BDL[1] | R<br>W | SBR7 | SBR6 | SBR5 | SBR4 | SBR3 | SBR2 | SBR1 | SBR0 |
| 0x0332 | SCI6CR1[1] | R<br>W | LOOPS | SCISWAI | RSRC | M | WAKE | ILT | PE | PT |
| 0x0330 | SCI6ASR1[(2)] | R<br>W | RXEDGIF | 0 | 0 | 0 | 0 | BERRV | BERRIF | BKDIF |
| 0x0331 | SCI6ACR1[2] | R<br>W | RXEDGIE | 0 | 0 | 0 | 0 | 0 | BERRIE | BKDIE |
| 0x0332 | SCI6ACR2[2] | R<br>W | 0 | 0 | 0 | 0 | 0 | BERRM1 | BERRM0 | BKDFE |
| 0x0333 | SCI6CR2 | R<br>W | TIE | TCIE | RIE | ILIE | TE | RE | RWU | SBK |
| 0x0334 | SCI6SR1 | R<br>W | TDRE | TC | RDRF | IDLE | OR | NF | FE | PF |

## 0x00330–0x0337 Asynchronous Serial Interface (SCI6) Map (continued)

| Address | Name | | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|---|---|
| 0x0335 | SCI6SR2 | R | AMAP | 0 | 0 | TXPOL | RXPOL | BRK13 | TXDIR | RAF |
| | | W | | | | | | | | |
| 0x0336 | SCI6DRH | R | R8 | T8 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | W | | | | | | | | |
| 0x0337 | SCI6DRL | R | R7 | R6 | R5 | R4 | R3 | R2 | R1 | R0 |
| | | W | T7 | T6 | T5 | T4 | T3 | T2 | T1 | T0 |

1. Those registers are accessible if the AMAP bit in the SCI6SR2 register is set to zero
2. Those registers are accessible if the AMAP bit in the SCI6SR2 register is set to one

## 0x00338–0x033F Asynchronous Serial Interface (SCI7) Map

| Address | Name | | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|---|---|
| 0x0338 | SCI7BDH[1] | R | IREN | TNP1 | TNP0 | SBR12 | SBR11 | SBR10 | SBR9 | SBR8 |
| | | W | | | | | | | | |
| 0x0339 | SCI7BDL[1] | R | SBR7 | SBR6 | SBR5 | SBR4 | SBR3 | SBR2 | SBR1 | SBR0 |
| | | W | | | | | | | | |
| 0x033A | SCI7CR1[1] | R | LOOPS | SCISWAI | RSRC | M | WAKE | ILT | PE | PT |
| | | W | | | | | | | | |
| 0x0338 | SCI7ASR1[2] | R | RXEDGIF | 0 | 0 | 0 | 0 | BERRV | BERRIF | BKDIF |
| | | W | | | | | | | | |
| 0x0339 | SCI7ACR1[2] | R | RXEDGIE | 0 | 0 | 0 | 0 | 0 | BERRIE | BKDIE |
| | | W | | | | | | | | |
| 0x033A | SCI7ACR2[2] | R | 0 | 0 | 0 | 0 | 0 | BERRM1 | BERRM0 | BKDFE |
| | | W | | | | | | | | |
| 0x033B | SCI7CR2 | R | TIE | TCIE | RIE | ILIE | TE | RE | RWU | SBK |
| | | W | | | | | | | | |
| 0x033C | SCI7SR1 | R | TDRE | TC | RDRF | IDLE | OR | NF | FE | PF |
| | | W | | | | | | | | |
| 0x033D | SCI7SR2 | R | AMAP | 0 | 0 | TXPOL | RXPOL | BRK13 | TXDIR | RAF |
| | | W | | | | | | | | |
| 0x033E | SCI7DRH | R | R8 | T8 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | W | | | | | | | | |
| 0x033F | SCI7DRL | R | R7 | R6 | R5 | R4 | R3 | R2 | R1 | R0 |
| | | W | T7 | T6 | T5 | T4 | T3 | T2 | T1 | T0 |

1. Those registers are accessible if the AMAP bit in the SCI7SR2 register is set to zero
2. Those registers are accessible if the AMAP bit in the SCI7SR2 register is set to one

## 0x00340–0x0367 – Periodic Interrupt Timer (PIT) Map (Sheet 1 of 3)

| Address | Name | | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|---|---|
| 0x0340 | PITCFLMT | R | PITE | PITSWAI | PITFRZ | 0 | 0 | 0 | 0 | 0 |
| | | W | | | | | | | PFLMT1 | PFLMT0 |
| 0x0341 | PITFLT | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | W | PFLT7 | PFLT6 | PFLT5 | PFLT4 | PFLT3 | PFLT2 | PFLT1 | PFLT0 |
| 0x0342 | PITCE | R | PCE7 | PCE6 | PCE5 | PCE4 | PCE3 | PCE2 | PCE1 | PCE0 |
| | | W | | | | | | | | |
| 0x0343 | PITMUX | R | PMUX7 | PMUX6 | PMUX5 | PMUX4 | PMUX3 | PMUX2 | PMUX1 | PMUX0 |
| | | W | | | | | | | | |
| 0x0344 | PITINTE | R | PINTE7 | PINTE6 | PINTE5 | PINTE4 | PINTE3 | PINTE2 | PINTE1 | PINTE0 |
| | | W | | | | | | | | |
| 0x0345 | PITTF | R | PTF7 | PTF6 | PTF5 | PTF4 | PTF3 | PTF2 | PTF1 | PTF0 |
| | | W | | | | | | | | |
| 0x0346 | PITMTLD0 | R | PMTLD7 | PMTLD6 | PMTLD5 | PMTLD4 | PMTLD3 | PMTLD2 | PMTLD1 | PMTLD0 |
| | | W | | | | | | | | |
| 0x0347 | PITMTLD1 | R | PMTLD7 | PMTLD6 | PMTLD5 | PMTLD4 | PMTLD3 | PMTLD2 | PMTLD1 | PMTLD0 |
| | | W | | | | | | | | |
| 0x0348 | PITLD0 (hi) | R | PLD15 | PLD14 | PLD13 | PLD12 | PLD11 | PLD10 | PLD9 | PLD8 |
| | | W | | | | | | | | |

## 0x00340–0x0367 – Periodic Interrupt Timer (PIT) Map (Sheet 2 of 3)

| Address | Name | | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|---|---|
| 0x0349 | PITLD0 (lo) | R<br>W | PLD7 | PLD6 | PLD5 | PLD4 | PLD3 | PLD2 | PLD1 | PLD0 |
| 0x034A | PITCNT0 (hi) | R<br>W | PCNT15 | PCNT14 | PCNT13 | PCNT12 | PCNT11 | PCNT10 | PCNT9 | PCNT8 |
| 0x034B | PITCNT0 (lo) | R<br>W | PCNT7 | PCNT6 | PCNT5 | PCNT4 | PCNT3 | PCNT2 | PCNT1 | PCNT0 |
| 0x034C | PITLD1 (hi) | R<br>W | PLD15 | PLD14 | PLD13 | PLD12 | PLD11 | PLD10 | PLD9 | PLD8 |
| 0x034D | PITLD1 (lo) | R<br>W | PLD7 | PLD6 | PLD5 | PLD4 | PLD3 | PLD2 | PLD1 | PLD0 |
| 0x034E | PITCNT1 (hi) | R<br>W | PCNT15 | PCNT14 | PCNT13 | PCNT12 | PCNT11 | PCNT10 | PCNT9 | PCNT8 |
| 0x034F | PITCNT1 (lo) | R<br>W | PCNT7 | PCNT6 | PCNT5 | PCNT4 | PCNT3 | PCNT2 | PCNT1 | PCNT0 |
| 0x0350 | PITLD2 (hi) | R<br>W | PLD15 | PLD14 | PLD13 | PLD12 | PLD11 | PLD10 | PLD9 | PLD8 |
| 0x0351 | PITLD2 (lo) | R<br>W | PLD7 | PLD6 | PLD5 | PLD4 | PLD3 | PLD2 | PLD1 | PLD0 |
| 0x0352 | PITCNT2 (hi) | R<br>W | PCNT15 | PCNT14 | PCNT13 | PCNT12 | PCNT11 | PCNT10 | PCNT9 | PCNT8 |
| 0x0353 | PITCNT2 (lo) | R<br>W | PCNT7 | PCNT6 | PCNT5 | PCNT4 | PCNT3 | PCNT2 | PCNT1 | PCNT0 |
| 0x0354 | PITLD3 (hi) | R<br>W | PLD15 | PLD14 | PLD13 | PLD12 | PLD11 | PLD10 | PLD9 | PLD8 |
| 0x0355 | PITLD3 (lo) | R<br>W | PLD7 | PLD6 | PLD5 | PLD4 | PLD3 | PLD2 | PLD1 | PLD0 |
| 0x0356 | PITCNT3 (hi) | R<br>W | PCNT15 | PCNT14 | PCNT13 | PCNT12 | PCNT11 | PCNT10 | PCNT9 | PCNT8 |
| 0x0357 | PITCNT3 (lo) | R<br>W | PCNT7 | PCNT6 | PCNT5 | PCNT4 | PCNT3 | PCNT2 | PCNT1 | PCNT0 |
| 0x0358 | PITLD4 (hi) | R<br>W | PLD15 | PLD14 | PLD13 | PLD12 | PLD11 | PLD10 | PLD9 | PLD8 |
| 0x0359 | PITLD4 (lo) | R<br>W | PLD7 | PLD6 | PLD5 | PLD4 | PLD3 | PLD2 | PLD1 | PLD0 |
| 0x035A | PITCNT4 (hi) | R<br>W | PCNT15 | PCNT14 | PCNT13 | PCNT12 | PCNT11 | PCNT10 | PCNT9 | PCNT8 |
| 0x035B | PITCNT4 (lo) | R<br>W | PCNT7 | PCNT6 | PCNT5 | PCNT4 | PCNT3 | PCNT2 | PCNT1 | PCNT0 |
| 0x035C | PITLD5 (hi) | R<br>W | PLD15 | PLD14 | PLD13 | PLD12 | PLD11 | PLD10 | PLD9 | PLD8 |
| 0x035D | PITLD5 (lo) | R<br>W | PLD7 | PLD6 | PLD5 | PLD4 | PLD3 | PLD2 | PLD1 | PLD0 |
| 0x035E | PITCNT5 (hi) | R<br>W | PCNT15 | PCNT14 | PCNT13 | PCNT12 | PCNT11 | PCNT10 | PCNT9 | PCNT8 |
| 0x035F | PITCNT5 (lo) | R<br>W | PCNT7 | PCNT6 | PCNT5 | PCNT4 | PCNT3 | PCNT2 | PCNT1 | PCNT0 |

## 0x00340–0x0367 – Periodic Interrupt Timer (PIT) Map (Sheet 3 of 3)

| Address | Name | | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|---|---|
| 0x0360 | PITLD6 (hi) | R<br>W | PLD15 | PLD14 | PLD13 | PLD12 | PLD11 | PLD10 | PLD9 | PLD8 |
| 0x0361 | PITLD6 (lo) | R<br>W | PLD7 | PLD6 | PLD5 | PLD4 | PLD3 | PLD2 | PLD1 | PLD0 |
| 0x0362 | PITCNT6 (hi) | R<br>W | PCNT15 | PCNT14 | PCNT13 | PCNT12 | PCNT11 | PCNT10 | PCNT9 | PCNT8 |
| 0x0363 | PITCNT6 (lo) | R<br>W | PCNT7 | PCNT6 | PCNT5 | PCNT4 | PCNT3 | PCNT2 | PCNT1 | PCNT0 |
| 0x0364 | PITLD7 (hi) | R<br>W | PLD15 | PLD14 | PLD13 | PLD12 | PLD11 | PLD10 | PLD9 | PLD8 |
| 0x0365 | PITLD7 (lo) | R<br>W | PLD7 | PLD6 | PLD5 | PLD4 | PLD3 | PLD2 | PLD1 | PLD0 |
| 0x0366 | PITCNT7 (hi) | R<br>W | PCNT15 | PCNT14 | PCNT13 | PCNT12 | PCNT11 | PCNT10 | PCNT9 | PCNT8 |
| 0x0367 | PITCNT7 (lo) | R<br>W | PCNT7 | PCNT6 | PCNT5 | PCNT4 | PCNT3 | PCNT2 | PCNT1 | PCNT0 |

## 0x0368–0x037F Port Integration Module (PIM) Map 6 of 6

| Address | Name | | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|---|---|
| 0x0368 | PTR | R | PTR7 | PTR6 | PTR5 | PTR4 | PTR3 | PTR2 | PTR1 | PTR0 |
| | | W | | | | | | | | |
| 0x0369 | PTIR | R | PTIR7 | PTIR6 | PTIR5 | PTIR4 | PTIR3 | PTIR2 | PTIR1 | PTIR0 |
| | | W | | | | | | | | |
| 0x036A | DDRR | R | DDRR7 | DDRR7 | DDRR5 | DDRR4 | DDRR3 | DDRR2 | DDRR1 | DDRR0 |
| | | W | | | | | | | | |
| 0x036B | RDRR | R | RDRR7 | RDRR6 | RDRR5 | RDRR4 | RDRR3 | RDRR2 | RDRR1 | RDRR0 |
| | | W | | | | | | | | |
| 0x036C | PERR | R | PERR7 | PERR6 | PERR5 | PERR4 | PERR3 | PERR2 | PERR1 | PERR0 |
| | | W | | | | | | | | |
| 0x036D | PPSR | R | PPSR7 | PPSR6 | PPSR5 | PPSR4 | PPSR3 | PPSR2 | PPSR1 | PPSR0 |
| | | W | | | | | | | | |
| 0x036E | Reserved | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | W | | | | | | | | |
| 0x036F | PTRRR | R | PTRRR7 | PTRRR6 | PTRRR5 | PTRRR4 | PTRRR3 | PTRRR2 | PTRRR1 | PTRRR0 |
| | | W | | | | | | | | |
| 0x0370 | PTL | R | PTL7 | PTL6 | PTL5 | PTL4 | PTL3 | PTL2 | PTL1 | PTL0 |
| | | W | | | | | | | | |
| 0x0371 | PTIL | R | PTIL7 | PTIL6 | PTIL5 | PTIL4 | PTIL3 | PTIL2 | PTIL1 | PTIL0 |
| | | W | | | | | | | | |
| 0x0372 | DDRL | R | DDRL7 | DDRL7 | DDRL5 | DDRL4 | DDRL3 | DDRL2 | DDRL1 | DDRL0 |
| | | W | | | | | | | | |
| 0x0373 | RDRL | R | RDRL7 | RDRL6 | RDRL5 | RDRL4 | RDRL3 | RDRL2 | RDRL1 | RDRL0 |
| | | W | | | | | | | | |
| 0x0374 | PERL | R | PERL7 | PERL6 | PERL5 | PERL4 | PERL3 | PERL2 | PERL1 | PERL0 |
| | | W | | | | | | | | |
| 0x0375 | PPSL | R | PPSL7 | PPSL6 | PPSL5 | PPSL4 | PPSL3 | PPSL2 | PPSL1 | PPSL0 |
| | | W | | | | | | | | |
| 0x0376 | WOML | R | WOML7 | WOML6 | WOML5 | WOML4 | WOML3 | WOML2 | WOML1 | WOML0 |
| | | W | | | | | | | | |
| 0x0377 | PTLRR | R | PTLRR7 | PTLRR6 | PTLRR5 | PTLRR4 | 0 | 0 | 0 | 0 |
| | | W | | | | | | | | |
| 0x0378 | PTF | R | PTF7 | PTF6 | PTF5 | PTF4 | PTF3 | PTF2 | PTF1 | PTF0 |
| | | W | | | | | | | | |
| 0x0379 | PTIF | R | PTIF7 | PTIF6 | PTIF5 | PTIF4 | PTIF3 | PTIF2 | PTIF1 | PTIF0 |
| | | W | | | | | | | | |
| 0x037A | DDRF | R | DDRF7 | DDRF7 | DDRF5 | DDRF4 | DDRF3 | DDRF2 | DDRF1 | DDRF0 |
| | | W | | | | | | | | |
| 0x037B | RDRF | R | RDRF7 | RDRF6 | RDRF5 | RDRF4 | RDRF3 | RDRF2 | RDRF1 | RDRF0 |
| | | W | | | | | | | | |
| 0x037C | PERF | R | PERF7 | PERF6 | PERF5 | PERF4 | PERF3 | PERF2 | PERF1 | PERF0 |
| | | W | | | | | | | | |

## 0x0368–0x037F Port Integration Module (PIM) Map 6 of 6 (continued)

| Address | Name | | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|---|---|
| 0x037D | PPSF | R | PPSF7 | PPSF6 | PPSF5 | PPSF4 | PPSF3 | PPSF2 | PPSF1 | PPSF0 |
| | | W | | | | | | | | |
| 0x037E | Reserved | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | W | | | | | | | | |
| 0x037F | PTFRR | R | 0 | 0 | PTFRR5 | PTFRR4 | PTFRR3 | PTFRR2 | PTFRR1 | PTFRR0 |
| | | W | | | | | | | | |

## 0x0380–0x03BF XGATE Map (Sheet 1 of 3)

| Address | Name | | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|---|---|
| 0x0380 | XGMCTL | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | XGIEM |
| | | W | XGEM | XGFRZM | XGDBGM | XGSSM | XGFACTM | | XGS WEFM | |
| 0x0381 | XGMCTL | R | XGE | XGFRZ | XGDBG | XGSS | XGFACT | 0 | XGSWEF | XGIE |
| | | W | | | | | | | | |
| 0x0382 | XGCHID | R | 0 | XGCHID[6:0] | | | | | | |
| | | W | | | | | | | | |
| 0x0383 | XGCHPL | R | 0 | 0 | 0 | 0 | 0 | XGCHPL | 0 | 0 |
| | | W | | | | | | | | |
| 0x0384 | Reserved | | | | | | | | | |
| | | | | | | | | | | |
| 0x0385 | XGISPSEL | R | 0 | 0 | 0 | 0 | 0 | 0 | XGISPSEL[1:0] | |
| | | W | | | | | | | | |
| 0x0386 | XGVBR | R | XGVBR[15:8] | | | | | | | |
| | | W | | | | | | | | |
| 0x0387 | XGVBR | R | XGVBR[7:1] | | | | | | | 0 |
| | | W | | | | | | | | |
| 0x0388 | XGIF | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | XGIF_78 |
| | | W | | | | | | | | |
| 0x0389 | XGIF | R | XGIF_77 | XGIF_76 | XGIF_75 | XGIF_74 | XGIF_73 | XGIF_72 | XGIF_71 | XGIF_70 |
| | | W | | | | | | | | |
| 0x038A | XGIF | R | XGIF_6F | XGIF_6E | XGIF_6D | XGIF_6C | XGIF_6B | XGIF_6A | XGIF_69 | XGIF_68 |
| | | W | | | | | | | | |
| 0x023B | XGIF | R | XGIF_67 | XGIF_66 | XGIF_65 | XGIF_64 | XGIF_63 | XGIF_62 | XGIF_61 | XGIF_60 |
| | | W | | | | | | | | |
| 0x023C | XGIF | R | XGIF_5F | XGIF_5E | XGIF_5D | XGIF_5C | XGIF_5B | XGIF_5A | XGIF_59 | XGIF_58 |
| | | W | | | | | | | | |
| 0x038D | XGIF | R | XGIF_57 | XGIF_56 | XGIF_55 | XGIF_54 | XGIF_53 | XGIF_52 | XGIF_51 | XGIF_50 |
| | | W | | | | | | | | |
| 0x038E | XGIF | R | XGIF_4F | XGIF_4E | XGIF_4D | XGIF_4C | XGIF_4B | XGIF_4A | XGIF_49 | XGIF_48 |
| | | W | | | | | | | | |
| 0x038F | XGIF | R | XGIF_47 | XGIF_46 | XGIF_45 | XGIF_44 | XGIF_43 | XGIF_42 | XGIF_41 | XGIF_40 |
| | | W | | | | | | | | |
| 0x0390 | XGIF | R | XGIF_3F | XGIF_3E | XGIF_3D | XGIF_3C | XGIF_3B | XGIF_3A | XGIF_39 | XGIF_38 |
| | | W | | | | | | | | |
| 0x0391 | XGIF | R | XGIF_37 | XGIF_36 | XGIF_35 | XGIF_34 | XGIF_33 | XGIF_32 | XGIF_31 | XGIF_30 |
| | | W | | | | | | | | |
| 0x0392 | XGIF | R | XGIF_2F | XGIF_2E | XGIF_2D | XGIF_2C | XGIF_2B | XGIF_2A | XGIF_29 | XGIF_28 |
| | | W | | | | | | | | |
| 0x0393 | XGIF | R | XGIF_27 | XGIF_26 | XGIF_25 | XGIF_24 | XGIF_23 | XGIF_22 | XGIF_21 | XGIF_20 |
| | | W | | | | | | | | |
| 0x0394 | XGIF | R | XGIF_1F | XGIF_1E | XGIF_1D | XGIF_1C | XGIF_1B | XGIF_1A | XGIF_19 | XGIF_18 |
| | | W | | | | | | | | |
| 0x0395 | XGIF | R | XGIF_17 | XGIF_16 | XGIF_15 | XGIF_14 | XGIF_13 | XGIF_12 | XGIF_11 | XGIF_10 |
| | | W | | | | | | | | |

## 0x0380–0x03BF XGATE Map (Sheet 2 of 3)

| Address | Name | | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|---|---|
| 0x0396 | XGIF | R | XGIF_0F | XGIF_0E | XGIF_0D | XGIF_0C | XGIF_0B | XGIF_0A | XGIF_09 | 0 |
| | | W | | | | | | | | |
| 0x0397 | XGIF | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | W | | | | | | | | |
| 0x0398 | XGSWTM | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | W | XGSWTM[7:0] | | | | | | | |
| 0x0399 | XGSWT | R | XGSWT[7:0] | | | | | | | |
| | | W | | | | | | | | |
| 0x039A | XGSEMM | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | W | XGSEMM[7:0] | | | | | | | |
| 0x039B | XGSEM | R | XGSEM[7:0] | | | | | | | |
| | | W | | | | | | | | |
| 0x039C | Reserved | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | W | | | | | | | | |
| 0x039D | XGCCR | R | 0 | 0 | 0 | 0 | XGN | XGZ | XGV | XGC |
| | | W | | | | | | | | |
| 0x039E | XGPC (hi) | R | XGPC[15:8] | | | | | | | |
| | | W | | | | | | | | |
| 0x039F | XGPC (lo) | R | XGPC[7:0] | | | | | | | |
| | | W | | | | | | | | |
| 0x03A0 | Reserved | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | W | | | | | | | | |
| 0x03A1 | Reserved | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | W | | | | | | | | |
| 0x03A2 | XGR1 (hi) | R | XGR1[15:8] | | | | | | | |
| | | W | | | | | | | | |
| 0x03A3 | XGR1 (lo) | R | XGR1[7:0] | | | | | | | |
| | | W | | | | | | | | |
| 0x03A4 | XGR2 (hi) | R | XGR2[15:8] | | | | | | | |
| | | W | | | | | | | | |
| 0x03A5 | XGR2 (lo) | R | XGR2[7:0] | | | | | | | |
| | | W | | | | | | | | |
| 0x03A6 | XGR3 (hi) | R | XGR3[15:8] | | | | | | | |
| | | W | | | | | | | | |
| 0x03A7 | XGR3 (lo) | R | XGR3[7:0] | | | | | | | |
| | | W | | | | | | | | |
| 0x03A8 | XGR4 (hi) | R | XGR4[15:8] | | | | | | | |
| | | W | | | | | | | | |
| 0x03A9 | XGR4 (lo) | R | XGR4[7:0] | | | | | | | |
| | | W | | | | | | | | |
| 0x03AA | XGR5 (hi) | R | XGR5[15:8] | | | | | | | |
| | | W | | | | | | | | |
| 0x03AB | XGR5(lo) | R | XGR5[7:0] | | | | | | | |
| | | W | | | | | | | | |
| 0x03AC | XGR6 (hi) | R | XGR6[15:8] | | | | | | | |
| | | W | | | | | | | | |

## 0x0380–0x03BF XGATE Map (Sheet 3 of 3)

| Address | Name | | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|---|---|
| 0x03AD | XGR6 (lo) | R | XGR6[7:0] | | | | | | | |
| | | W | | | | | | | | |
| 0x03AE | XGR7 (hi) | R | XGR7[15:8] | | | | | | | |
| | | W | | | | | | | | |
| 0x03AF | XGR7 (lo) | R | XGR7[7:0] | | | | | | | |
| | | W | | | | | | | | |
| 0x03B0–0x03BF | Reserved | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | W | | | | | | | | |

## 0x03C0–0x03CF Reserved

| Address | Name | | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|---|---|
| 0x03C0 -0x03CF | Reserved | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | W | | | | | | | | |

## 0x03D0–0x03FF Timer Module (TIM) Map (Sheet 1 of 2)

| Address | Name | | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|---|---|
| 0x03D0 | TIOS | R | IOS7 | IOS6 | IOS5 | IOS4 | IOS3 | IOS2 | IOS1 | IOS0 |
| | | W | | | | | | | | |
| 0x03D1 | CFORC | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | W | FOC7 | FOC6 | FOC5 | FOC4 | FOC3 | FOC2 | FOC1 | FOC0 |
| 0x03D2 | OC7M | R | OC7M7 | OC7M6 | OC7M5 | OC7M4 | OC7M3 | OC7M2 | OC7M1 | OC7M0 |
| | | W | | | | | | | | |
| 0x03D3 | OC7D | R | OC7D7 | OC7D6 | OC7D5 | OC7D4 | OC7D3 | OC7D2 | OC7D1 | OC7D0 |
| | | W | | | | | | | | |
| 0x03D4 | TCNTH | R | TCNT15 | TCNT14 | TCNT13 | TCNT12 | TCNT11 | TCNT10 | TCNT9 | TCNT8 |
| | | W | | | | | | | | |
| 0x03D5 | TCNTL | R | TCNT7 | TCNT6 | TCNT5 | TCNT4 | TCNT3 | TCNT2 | TCNT1 | TCNT0 |
| | | W | | | | | | | | |
| 0x03D6 | TSCR1 | R | TEN | TSWAI | TSFRZ | TFFCA | PRNT | 0 | 0 | 0 |
| | | W | | | | | | | | |
| 0x03D7 | TTOV | R | TOV7 | TOV6 | TOV5 | TOV4 | TOV3 | TOV2 | TOV1 | TOV0 |
| | | W | | | | | | | | |
| 0x03D8 | TCTL1 | R | OM7 | OL7 | OM6 | OL6 | OM5 | OL5 | OM4 | OL4 |
| | | W | | | | | | | | |
| 0x03D9 | TCTL2 | R | OM3 | OL3 | OM2 | OL2 | OM1 | OL1 | OM0 | OL0 |
| | | W | | | | | | | | |
| 0x03DA | TCTL3 | R | EDG7B | EDG7A | EDG6B | EDG6A | EDG5B | EDG5A | EDG4B | EDG4A |
| | | W | | | | | | | | |
| 0x03DB | TCTL4 | R | EDG3B | EDG3A | EDG2B | EDG2A | EDG1B | EDG1A | EDG0B | EDG0A |
| | | W | | | | | | | | |
| 0x03DC | TIE | R | C7I | C6I | C5I | C4I | C3I | C2I | C1I | C0I |
| | | W | | | | | | | | |
| 0x03DD | TSCR2 | R | TOI | 0 | 0 | 0 | TCRE | PR2 | PR1 | PR0 |
| | | W | | | | | | | | |
| 0x03DE | TFLG1 | R | C7F | C6F | C5F | C4F | C3F | C2F | C1F | C0F |
| | | W | | | | | | | | |
| 0x03DF | TFLG2 | R | TOF | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | W | | | | | | | | |
| 0x03E0 | TC0H | R | Bit 15 | Bit 14 | Bit 13 | Bit 12 | Bit 11 | Bit 10 | Bit 9 | Bit 8 |
| | | W | | | | | | | | |
| 0x03E1 | TC0L | R | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
| | | W | | | | | | | | |
| 0x03E2 | TC1H | R | Bit 15 | Bit 14 | Bit 13 | Bit 12 | Bit 11 | Bit 10 | Bit 9 | Bit 8 |
| | | W | | | | | | | | |
| 0x03E3 | TC1L | R | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
| | | W | | | | | | | | |

## 0x03D0–0x03FF Timer Module (TIM) Map (Sheet 2 of 2)

| Address | Name | | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|---|---|
| 0x03E4 | TC2H | R<br>W | Bit 15 | Bit 14 | Bit 13 | Bit 12 | Bit 11 | Bit 10 | Bit 9 | Bit 8 |
| 0x03E5 | TC2L | R<br>W | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
| 0x03E6 | TC3H | R<br>W | Bit 15 | Bit 14 | Bit 13 | Bit 12 | Bit 11 | Bit 10 | Bit 9 | Bit 8 |
| 0x03E7 | TC3L | R<br>W | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
| 0x03E8 | TC4H | R<br>W | Bit 15 | Bit 14 | Bit 13 | Bit 12 | Bit 11 | Bit 10 | Bit 9 | Bit 8 |
| 0x03E9 | TC4L | R<br>W | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
| 0x03EA | TC5H | R<br>W | Bit 15 | Bit 14 | Bit 13 | Bit 12 | Bit 11 | Bit 10 | Bit 9 | Bit 8 |
| 0x03EB | TC5L | R<br>W | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
| 0x03EC | TC6H | R<br>W | Bit 15 | Bit 14 | Bit 13 | Bit 12 | Bit 11 | Bit 10 | Bit 9 | Bit 8 |
| 0x03ED | TC6L | R<br>W | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
| 0x03EE | TC7H | R<br>W | Bit 15 | Bit 14 | Bit 13 | Bit 12 | Bit 11 | Bit 10 | Bit 9 | Bit 8 |
| 0x03EF | TC7L | R<br>W | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
| 0x03F0 | PACTL | R<br>W | 0 | PAEN | PAMOD | PEDGE | CLK1 | CLK0 | PAOVI | PAI |
| 0x03F1 | PAFLG | R<br>W | 0 | 0 | 0 | 0 | 0 | 0 | PAOVF | PAIF |
| 0x03F2 | PACNTH | R<br>W | PACNT15 | PACNT14 | PACNT13 | PACNT12 | PACNT11 | PACNT10 | PACNT9 | PACNT8 |
| 0x03F3 | PACNTL | R<br>W | PACNT7 | PACNT6 | PACNT5 | PACNT4 | PACNT3 | PACNT2 | PACNT1 | PACNT0 |
| 0x03F4–<br>0x03FB | Reserved | R<br>W | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0x03FC | OCPD | R<br>W | OCPD7 | OCPD6 | OCPD5 | OCPD4 | OCPD3 | OCPD2 | OCPD1 | OCPD0 |
| 0x03FD | Reserved | R<br>W | | | | | | | | |
| 0x03FE | PTPSR | R<br>W | PTPSR7 | PTPSR6 | PTPSR5 | PTPSR4 | PTPSR3 | PTPSR2 | PTPSR1 | PTPSR0 |
| 0x03FF | Reserved | R<br>W | | | | | | | | |

## 0x0400–0x07FF Reserved

| Address | Name | | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|---|---|
| 0x0400–0x07FF | Reserved | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | W | | | | | | | | |

# Appendix F
# Ordering Information

Customers can choose between ordering either the mask-specific partnumber or the generic / mask-independent partnumber. Ordering the mask-specific partnumber enables the customer to specify which particular maskset they receive whereas ordering the generic maskset means that the currently preferred maskset (which may change over time) is shipped. In either case, the marking on the device will always show the generic / mask-independent partnumber and the mask set number.

**NOTE**

**The mask identifier suffix and the Tape & Reel suffix are always both omitted from the partnumber which is actually marked on the device.**

For specific partnumbers to order, please contact your local sales office. The below figure illustrates the structure of a typical mask-specific ordering number for the MC9S12XE-Family devices

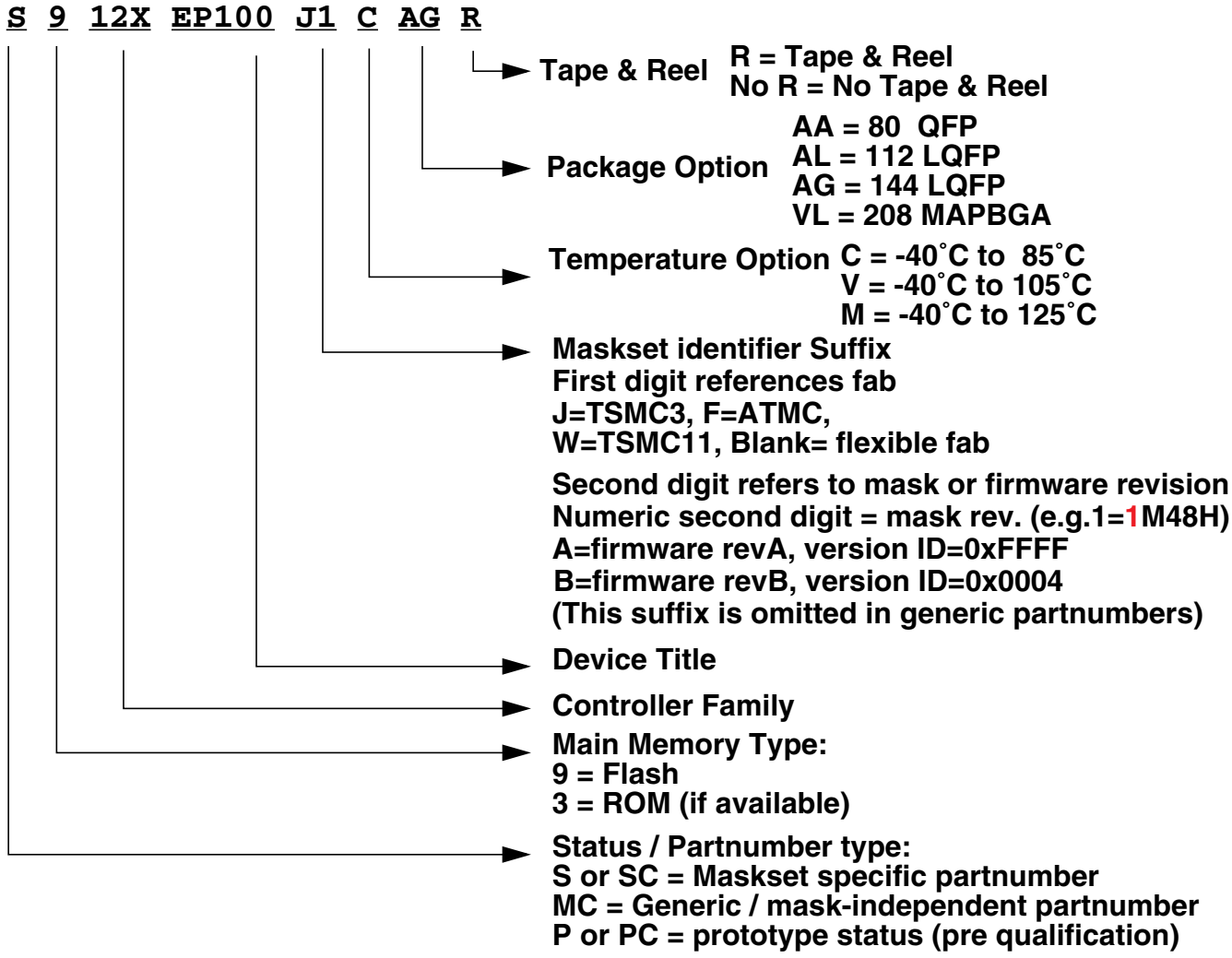



**Figure F-1. Order Part Number Example**

***How to Reach Us:***

**Home Page:**
www.freescale.com

**USA/Europe or Locations Not Listed:**
Freescale Semiconductor
Technical Information Center, CH370
1300 N. Alma School Road
Chandler, Arizona 85224
1-800-521-6274 or 480-768-2130

**Europe, Middle East, and Africa:**
+44 1296 380 456 (English)
+46 8 52200080 (English)
+49 89 92103 559 (German)
+33 1 69 35 48 48 (French)

**Japan:**
Freescale Semiconductor Japan Ltd.
Technical Information Center
3-20-1, Minami-Azabu, Minato-ku
Tokyo 106-0047, Japan
0120-191014 or +81-3-3440-3569

**Asia/Pacific:**
Freescale Semiconductor Hong Kong Ltd.
Technical Information Center
2 Dai King Street
Tai Po Industrial Estate
Tai Po, N.T., Hong Kong
852-26668334

***For Literature Requests Only:***
Freescale Semiconductor Literature Distribution Center
P.O. Box 5405
Denver, Colorado 80217
1-800-441-2447 or 303-675-2140
Fax: 303-675-2150

Information in this document is provided solely to enable system and software implementers to use Freescale Semiconductor products. There are no express or implied copyright licenses granted hereunder to design or fabricate any integrated circuits or integrated circuits based on the information in this document.

Freescale Semiconductor reserves the right to make changes without further notice to any products herein. Freescale Semiconductor makes no warranty, representation or guarantee regarding the suitability of its products for any particular purpose, nor does Freescale Semiconductor assume any liability arising out of the application or use of any product or circuit, and specifically disclaims any and all liability, including without limitation consequential or incidental damages. “Typical” parameters that may be provided in Freescale Semiconductor data sheets and/or specifications can and do vary in different applications and actual performance may vary over time. All operating parameters, including “Typicals”, must be validated for each customer application by customer’s technical experts. Freescale Semiconductor does not convey any license under its patent rights nor the rights of others. Freescale Semiconductor products are not designed, intended, or authorized for use as components in systems intended for surgical implant into the body, or other applications intended to support or sustain life, or for any other application in which the failure of the Freescale Semiconductor product could create a situation where personal injury or death may occur. Should Buyer purchase or use Freescale Semiconductor products for any such unintended or unauthorized application, Buyer shall indemnify and hold Freescale Semiconductor and its officers, employees, subsidiaries, affiliates, and distributors harmless against all claims, costs, damages, and expenses, and reasonable attorney fees arising out of, directly or indirectly, any claim of personal injury or death associated with such unintended or unauthorized use, even if such claim alleges that Freescale Semiconductor was negligent regarding the design or manufacture of the part.

Freescale™ and the Freescale logo are trademarks of Freescale Semiconductor, Inc. All other product or service names are the property of their respective owners.

© Freescale Semiconductor, Inc. 2005-2013. All rights reserved.

MC9S12XEP100RMV1
Rev. 1.25
02/2013