MWCT10x3A Reference Manual

Supports MWCT1003AVLH, MWCT1013AVLH

Document Number: MWCT10x3ARM
Rev. 2, 09/2016

Contents

Section number **Title** **Page**

Chapter 1
About This Document

1.1 Overview....45

1.1.1 Purpose....45

1.1.2 Audience....45

1.2 Conventions....45

1.2.1 Numbering systems....45

1.2.2 Typographic notation....46

1.2.3 Special terms....46

Chapter 2
Introduction

2.1 Introduction....47

2.1.1 Core Overview....47

2.1.2 Memory Overview....48

2.1.3 Peripheral Overview....48

2.2 Application Examples....48

2.3 Features....48

2.3.1 MWCT10x3A product features....49

2.3.2 Block Diagram....50

2.3.3 56800EX 32-bit core....51

2.3.4 Operation Parameters....52

2.3.5 Packages....53

2.3.6 On-Chip Memory and Memory Protection....53

2.3.7 Peripherals....53

2.3.7.1 System Modules....53

2.3.7.2 General Purpose I/O (GPIO)....55

2.3.7.3 Timers and PWM modules....55

2.3.7.4 Clock Modules....57

2.3.7.5 Analog Modules....58
2.3.7.6 Communication Interfaces....59
2.3.7.7 Power Management....61

Chapter 3
Chip Configuration

3.1 Introduction....63
3.2 Core Configuration....63
3.3 System modules....64
3.3.1 System Integration Module (SIM) Configuration....64
3.3.2 MCM Configuration....64
3.3.3 Inter-Peripheral Crossbar Switch (XBAR) and AND/OR/INVERT (AOI) Configuration....65
3.3.3.1 Number of inputs and outputs....66
3.3.3.2 XBARA and XBARB Inputs....66
3.3.3.3 XBAR Interconnections....67
3.3.3.4 XBARA Outputs....68
3.3.3.5 AOI module register write protection....70
3.3.4 Interrupt Controller (INTC) Configuration....70
3.3.4.1 Reset/Interrupt Vector Table....71
3.3.5 DMA Controller Configuration....79
3.3.5.1 DMA channel assignments....79
3.3.6 Power Management Controller (PMC) Configuration....80
3.4 Clock Modules....81
3.4.1 On-Chip Clock Synthesis (OCCS) Configuration....81
3.5 Memories and Memory Interfaces....82
3.5.1 Flash Memory Controller (FMC) Configuration....82
3.5.2 Flash Memory Configuration....83
3.5.2.1 Flash memory types and terminology....84
3.5.2.2 FTFL_FOPT Register....84
3.6 Security and Integrity....84

3.6.1 Computer Operating Properly (COP) Module Configuration....84
3.6.1.1 COP low power clocks....85
3.6.2 External Watchdog Monitor (EWM) Configuration....85
3.6.2.1 EWM low power clocks....86
3.6.2.2 EWM_OUT pin state in Low Power Modes....86
3.6.2.3 EWM_IN signal....87
3.6.3 Cyclic Redundancy Check (CRC) Configuration....87
3.7 Analog....87
3.7.1 SAR Analog-to-Digital Converter (ADC) Configuration....87
3.7.1.1 SAR ADC Instantiation....88
3.7.1.2 SAR ADC Channel Assignments....88
3.7.1.3 SAR ADC clock in stop modes....89
3.7.1.4 SAR ADC alternate clock....89
3.7.1.5 SAR ADC Reference Options....89
3.7.1.6 SAR ADC Hardware Conversion Trigger Source....90
3.7.2 Cyclic Analog-to-Digital Converter (ADC) Configuration....90
3.7.2.1 Cyclic ADC Instantiation....90
3.7.2.2 Cyclic ADC SYNC Signal Connections....91
3.7.2.3 Cyclic ADC and PWM Connections....91
3.7.3 Comparator (CMP) Configuration....91
3.7.3.1 Comparator Channel Assignments....92
3.7.3.2 Comparator Voltage References....93
3.7.4 12-bit Digital-to-Analog Converter (DAC) Configuration....93
3.8 Timers and PWM....94
3.8.1 PWM Configuration....94
3.8.1.1 PWM auxiliary signals and analog inputs....95
3.8.2 PDB Configuration....95
3.8.2.1 PDB Trigger Connections....96
3.8.3 PIT Configuration....97

3.8.3.1 PIT low power clocks ... 97
3.8.3.2 PIT master/slave selection ... 98
3.8.4 TMR Configuration ... 98
3.9 Communication interfaces ... 99
3.9.1 CAN Configuration ... 99
3.9.1.1 FlexCAN3 glitch filter ... 99
3.9.1.2 FlexCAN3 Supervisor Mode ... 100
3.9.2 Serial Peripheral Interface (SPI) Configuration ... 100
3.9.3 Inter-Integrated Circuit (I2C) Configuration ... 101
3.9.3.1 I2C module address matching to wake the device from stop mode ... 102
3.9.4 SCI Configuration ... 102
3.10 Human-machine interfaces (HMI) ... 103
3.10.1 GPIO Configuration ... 103
3.10.1.1 GPIO Port D[4:0] configuration ... 104
3.10.1.2 GPIO unbonded pads ... 105

Chapter 4
Memory Map

4.1 Introduction ... 107
4.2 Program/Data Memory Maps ... 107
4.3 Core and System Peripheral Memory Map ... 108
4.4 Slave Peripheral Memory Map ... 109

Chapter 5
Clock Distribution

5.1 Overview ... 113
5.2 Clock definitions ... 113
5.3 System clock source configuration ... 114
5.4 Module clocks ... 115

Chapter 6
Reset and Boot

6.1 Reset Configuration ... 117

6.2 System Boot....118
6.2.1 FOPT boot options....118
6.2.2 Boot Procedure for Normal Operation....118
6.2.3 Boot sequence....119

Chapter 7
Power Management

7.1 Overview....121
7.2 Architecture....121
7.3 External Supplies and Regulation....122
7.4 User Power Management Methods....122
7.5 Power Modes....123
7.6 Power mode transitions....125

Chapter 8
Signal Multiplexing and Signal Descriptions

8.1 Introduction....129
8.2 Signal Multiplexing Integration....129
8.3 Signal Multiplexing and Pin Assignments....130
8.4 Pinout diagrams....132

Chapter 9
Memory Resource Protection (MRP)

9.1 Overview....135
9.2 Features....136
9.3 Operation....136
9.4 Programming Model Overview....140
9.5 Memory Resource Protection Restrictions....140
9.6 Base Address Setup....140
9.7 Programming Example....142

Chapter 10
Miscellaneous Control Module (MCM)

10.1 Introduction....145

10.1.1 Features ... 145
10.2 Memory Map/Register Descriptions ... 146
10.2.1 Crossbar switch (AXBS) slave configuration (MCM_PLASC) ... 147
10.2.2 Crossbar switch (AXBS) master configuration (MCM_PLAMC) ... 147
10.2.3 Core control register (MCM_CPCR) ... 148
10.2.4 Core fault address register (MCM_CFADR) ... 149
10.2.5 Core fault attributes register (MCM_CFATR) ... 150
10.2.6 Core fault location register (MCM_CFLOC) ... 151
10.2.7 Core fault interrupt enable register (MCM_CFIER) ... 152
10.2.8 MCM interrupt status register (MCM_CFISR) ... 152
10.2.9 Core fault data register (MCM_CFDTR) ... 153
10.2.10 Resource Protection Control Register (MCM_RPCR) ... 153
10.2.11 User Flash Base Address Register (MCM_UFLASHBAR) ... 155
10.2.12 User Program RAM Base Address Register (MCM_UPRAMBAR) ... 155
10.2.13 Resource Protection Other Stack Pointer (MCM_SRPOSP) ... 156
10.2.14 Memory Protection Illegal PC (MCM_SRPIPC) ... 156
10.2.15 Resource Protection Misaligned PC (MCM_SRPMPC) ... 158
10.3 Functional Description ... 159
10.3.1 Core Data Fault Recovery Registers ... 159

Chapter 11
System Integration Module (SIM)

11.1 Introduction ... 161
11.1.1 Overview ... 161
11.1.2 Features ... 161
11.1.3 Modes of Operation ... 162
11.1.4 Block Diagram ... 163
11.2 Memory Map and Register Descriptions ... 165
11.2.1 Control Register (SIM_CTRL) ... 166
11.2.2 Reset Status Register (SIM_RSTAT) ... 168

11.2.3 Software Control Register (SIM_SCR0)..... 169
11.2.4 Software Control Register (SIM_SCR1)..... 170
11.2.5 Software Control Register (SIM_SCR2)..... 170
11.2.6 Software Control Register (SIM_SCR3)..... 170
11.2.7 Most Significant Half of JTAG ID (SIM_MSHID)..... 171
11.2.8 Least Significant Half of JTAG ID (SIM_LSHID)..... 172
11.2.9 Power Control Register (SIM_PWR)..... 173
11.2.10 Clock Output Select Register (SIM_CLKOUT)..... 174
11.2.11 Peripheral Clock Rate Register (SIM_PCR)..... 176
11.2.12 Peripheral Clock Enable Register 0 (SIM_PCE0)..... 177
11.2.13 Peripheral Clock Enable Register 1 (SIM_PCE1)..... 179
11.2.14 Peripheral Clock Enable Register 2 (SIM_PCE2)..... 181
11.2.15 Peripheral Clock Enable Register 3 (SIM_PCE3)..... 182
11.2.16 STOP Disable Register 0 (SIM_SD0)..... 183
11.2.17 Peripheral Clock STOP Disable Register 1 (SIM_SD1)..... 186
11.2.18 Peripheral Clock STOP Disable Register 2 (SIM_SD2)..... 188
11.2.19 Peripheral Clock STOP Disable Register 3 (SIM_SD3)..... 190
11.2.20 I/O Short Address Location Register (SIM_IOSAHI)..... 192
11.2.21 I/O Short Address Location Register (SIM_IOSALO)..... 193
11.2.22 Protection Register (SIM_PROT)..... 194
11.2.23 GPIOA LSBs Peripheral Select Register (SIM_GPSAL)..... 196
11.2.24 GPIOC LSBs Peripheral Select Register (SIM_GPSCL)..... 197
11.2.25 GPIOC MSBs Peripheral Select Register (SIM_GPSCH)..... 198
11.2.26 GPIOE LSBs Peripheral Select Register (SIM_GPSEL)..... 200
11.2.27 GPIOF LSBs Peripheral Select Register (SIM_GPSFL)..... 201
11.2.28 GPIOF MSBs Peripheral Select Register (SIM_GPSFH)..... 202
11.2.29 Internal Peripheral Select Register 0 (SIM_IPS0)..... 202
11.2.30 Miscellaneous Register 0 (SIM_MISC0)..... 205
11.2.31 Peripheral Software Reset Register 0 (SIM_PSWR0)..... 206

11.2.32 Peripheral Software Reset Register 1 (SIM_PSWR1)..... 207
11.2.33 Peripheral Software Reset Register 2 (SIM_PSWR2)..... 208
11.2.34 Peripheral Software Reset Register 3 (SIM_PSWR3)..... 210
11.2.35 Power Mode Register (SIM_PWRMODE)..... 211
11.2.36 Non-Volatile Memory Option Register 2 (High) (SIM_NVMOPT2H)..... 212
11.2.37 Non-Volatile Memory Option Register 2 (Low) (SIM_NVMOPT2L)..... 213
11.2.38 Non-Volatile Memory Option Register 3 (High) (SIM_NVMOPT3H)..... 213
11.3 Functional Description..... 214
11.3.1 Clock Generation Overview..... 214
11.3.2 Power-Down Modes Overview..... 215
11.3.3 STOP and WAIT Mode Disable Function..... 216
11.4 Resets..... 217
11.5 Clocks..... 217
11.6 Interrupts..... 217

Chapter 12
Interrupt Controller (INTC)

12.1 Introduction..... 219
12.1.1 References..... 219
12.1.2 Features..... 219
12.1.3 Modes of Operation..... 219
12.1.4 Block Diagram..... 220
12.2 Memory Map and Registers..... 221
12.2.1 Interrupt Priority Register 0 (INTC_IPR0)..... 222
12.2.2 Interrupt Priority Register 1 (INTC_IPR1)..... 223
12.2.3 Interrupt Priority Register 2 (INTC_IPR2)..... 225
12.2.4 Interrupt Priority Register 3 (INTC_IPR3)..... 226
12.2.5 Interrupt Priority Register 4 (INTC_IPR4)..... 228
12.2.6 Interrupt Priority Register 5 (INTC_IPR5)..... 229
12.2.7 Interrupt Priority Register 6 (INTC_IPR6)..... 231

12.2.8 Interrupt Priority Register 8 (INTC_IPR8)....232
12.2.9 Interrupt Priority Register 9 (INTC_IPR9)....233
12.2.10 Interrupt Priority Register 10 (INTC_IPR10)....234
12.2.11 Interrupt Priority Register 11 (INTC_IPR11)....236
12.2.12 Interrupt Priority Register 12 (INTC_IPR12)....237
12.2.13 Vector Base Address Register (INTC_VBA)....238
12.2.14 Fast Interrupt 0 Match Register (INTC_FIM0)....239
12.2.15 Fast Interrupt 0 Vector Address Low Register (INTC_FIVAL0)....239
12.2.16 Fast Interrupt 0 Vector Address High Register (INTC_FIVAH0)....240
12.2.17 Fast Interrupt 1 Match Register (INTC_FIM1)....240
12.2.18 Fast Interrupt 1 Vector Address Low Register (INTC_FIVAL1)....241
12.2.19 Fast Interrupt 1 Vector Address High Register (INTC_FIVAH1)....241
12.2.20 IRQ Pending Register 0 (INTC_IRQP0)....241
12.2.21 IRQ Pending Register 1 (INTC_IRQP1)....242
12.2.22 IRQ Pending Register 2 (INTC_IRQP2)....242
12.2.23 IRQ Pending Register 3 (INTC_IRQP3)....243
12.2.24 IRQ Pending Register 4 (INTC_IRQP4)....243
12.2.25 IRQ Pending Register 5 (INTC_IRQP5)....244
12.2.26 IRQ Pending Register 6 (INTC_IRQP6)....244
12.2.27 Control Register (INTC_CTRL)....245
12.3 Functional Description....246
12.3.1 Normal Interrupt Handling....246
12.3.2 Interrupt Nesting....246
12.3.3 Fast Interrupt Handling....247
12.4 Interrupts....247

Chapter 13
DMA Controller

13.1 Introduction....249
13.1.1 Overview....249

13.1.2 Features ... 251
13.2 DMA Transfer Overview ... 251
13.3 Memory Map/Register Definition ... 253
13.3.1 DMA Request Control Register (DMA_REQC) ... 254
13.3.2 Source Address Register (DMA_SAR*n*) ... 258
13.3.3 Destination Address Register (DMA_DAR*n*) ... 258
13.3.4 DMA Status Register / Byte Count Register (DMA_DSR_BCR*n*) ... 258
13.3.5 DMA Control Register (DMA_DCR*n*) ... 261
13.4 Functional Description ... 264
13.4.1 Transfer requests (Cycle-Steal and Continuous modes) ... 265
13.4.2 Channel initialization and startup ... 265
13.4.2.1 Channel prioritization ... 265
13.4.2.2 Programming the DMA Controller Module ... 266
13.4.3 Dual-Address Data Transfer Mode ... 267
13.4.4 Advanced Data Transfer Controls: Auto-Alignment ... 268
13.4.5 Termination ... 269

Chapter 14
Power Management Controller (PMC)

14.1 Introduction ... 271
14.1.1 Overview ... 271
14.1.2 Features ... 271
14.1.3 Modes of Operation ... 272
14.1.4 Block Diagram ... 273
14.2 Memory Map and Register Descriptions ... 274
14.2.1 Control Register (PMC_CTRL) ... 275
14.2.2 Status Register (PMC_STS) ... 276
14.3 Functional Description ... 278
14.4 Resets ... 279
14.5 Clocks ... 279

14.6 Interrupts....280

Chapter 15
Crossbar AND/OR/INVERT (AOI) Module

15.1 Introduction....281
15.1.1 Overview....281
15.1.2 Features....283
15.1.3 Modes of Operation....283
15.2 External Signal Description....283
15.3 Memory Map and Register Descriptions....283
15.3.1 Boolean Function Term 0 and 1 Configuration Register for EVENTn (AOI_BFCRT01*n*)....285
15.3.2 Boolean Function Term 2 and 3 Configuration Register for EVENTn (AOI_BFCRT23*n*)....286
15.4 Functional Description....288
15.4.1 Configuration Examples for the Boolean Function Evaluation....289
15.4.2 AOI Timing Between Inputs and Outputs....290

Chapter 16
Inter-Peripheral Crossbar Switch A (XBARA)

16.1 Introduction....293
16.1.1 Overview....293
16.1.2 Features....293
16.1.3 Modes of Operation....294
16.1.4 Block Diagram....294
16.2 Signal Descriptions....295
16.2.1 XBAR_OUT[0:NUM_OUT-1] - MUX Outputs....295
16.2.2 XBAR_IN[0:NUM_IN-1] - MUX Inputs....295
16.2.3 DMA_REQ[n] - DMA Request Output(s)....295
16.2.4 DMA_ACK[n] - DMA Acknowledge Input(s)....295
16.2.5 INT_REQ[n] - Interrupt Request Output(s)....296
16.3 Memory Map and Register Descriptions....296
16.3.1 Crossbar A Select Register 0 (XBARA_SEL0)....297

16.3.2 Crossbar A Select Register 1 (XBARA_SEL1)....298
16.3.3 Crossbar A Select Register 2 (XBARA_SEL2)....298
16.3.4 Crossbar A Select Register 3 (XBARA_SEL3)....299
16.3.5 Crossbar A Select Register 4 (XBARA_SEL4)....299
16.3.6 Crossbar A Select Register 5 (XBARA_SEL5)....300
16.3.7 Crossbar A Select Register 6 (XBARA_SEL6)....300
16.3.8 Crossbar A Select Register 7 (XBARA_SEL7)....301
16.3.9 Crossbar A Select Register 8 (XBARA_SEL8)....301
16.3.10 Crossbar A Select Register 9 (XBARA_SEL9)....302
16.3.11 Crossbar A Select Register 10 (XBARA_SEL10)....302
16.3.12 Crossbar A Select Register 11 (XBARA_SEL11)....303
16.3.13 Crossbar A Select Register 12 (XBARA_SEL12)....303
16.3.14 Crossbar A Select Register 13 (XBARA_SEL13)....304
16.3.15 Crossbar A Select Register 14 (XBARA_SEL14)....304
16.3.16 Crossbar A Select Register 15 (XBARA_SEL15)....305
16.3.17 Crossbar A Select Register 16 (XBARA_SEL16)....305
16.3.18 Crossbar A Select Register 17 (XBARA_SEL17)....306
16.3.19 Crossbar A Select Register 18 (XBARA_SEL18)....306
16.3.20 Crossbar A Select Register 19 (XBARA_SEL19)....307
16.3.21 Crossbar A Select Register 20 (XBARA_SEL20)....307
16.3.22 Crossbar A Select Register 21 (XBARA_SEL21)....308
16.3.23 Crossbar A Select Register 22 (XBARA_SEL22)....308
16.3.24 Crossbar A Select Register 23 (XBARA_SEL23)....309
16.3.25 Crossbar A Select Register 24 (XBARA_SEL24)....309
16.3.26 Crossbar A Select Register 25 (XBARA_SEL25)....310
16.3.27 Crossbar A Select Register 26 (XBARA_SEL26)....310
16.3.28 Crossbar A Select Register 27 (XBARA_SEL27)....311
16.3.29 Crossbar A Select Register 28 (XBARA_SEL28)....311
16.3.30 Crossbar A Select Register 29 (XBARA_SEL29)....312

16.3.31 Crossbar A Control Register 0 (XBARA_CTRL0)..... 312
16.3.32 Crossbar A Control Register 1 (XBARA_CTRL1)..... 314
16.4 Functional Description.....316
16.4.1 General..... 316
16.4.2 Functional Mode..... 317
16.5 Resets..... 317
16.6 Clocks..... 317
16.7 Interrupts and DMA Requests..... 317

Chapter 17
Inter-Peripheral Crossbar Switch B (XBARB): AOI Input

17.1 Introduction.....319
17.2 Memory Map and Register Descriptions..... 319
17.2.1 Crossbar B Select Register 0 (XBARB_SEL0)..... 320
17.2.2 Crossbar B Select Register 1 (XBARB_SEL1)..... 320
17.2.3 Crossbar B Select Register 2 (XBARB_SEL2)..... 321
17.2.4 Crossbar B Select Register 3 (XBARB_SEL3)..... 321
17.2.5 Crossbar B Select Register 4 (XBARB_SEL4)..... 322
17.2.6 Crossbar B Select Register 5 (XBARB_SEL5)..... 322
17.2.7 Crossbar B Select Register 6 (XBARB_SEL6)..... 323
17.2.8 Crossbar B Select Register 7 (XBARB_SEL7)..... 323

Chapter 18
On-Chip Clock Synthesis (OCCS)

18.1 Introduction.....325
18.1.1 Overview..... 325
18.1.2 Features..... 325
18.2 Modes of Operation..... 326
18.2.1 Internal Clock Sources..... 327
18.2.2 Loop Controlled Pierce Crystal Oscillator.....327
18.2.3 External Clock Source - Crystal Oscillator Bypass Option..... 328

18.2.4 External Clock Source - CLKIN....328
18.3 Block Diagram....330
18.4 Pin Description....331
18.4.1 External Clock Reference....331
18.4.2 Oscillator IO (XTAL, EXTAL)....331
18.4.3 CLKO....331
18.5 Memory Map and Register Descriptions....332
18.5.1 PLL Control Register (OCCS_CTRL)....332
18.5.2 PLL Divide-By Register (OCCS_DIVBY)....334
18.5.3 OCCS Status Register (OCCS_STAT)....335
18.5.4 Oscillator Control Register 1 (OCCS_OSCTL1)....337
18.5.5 Oscillator Control Register 2 (OCCS_OSCTL2)....338
18.5.6 External Clock Check Reference (OCCS_CLKCHKR)....340
18.5.7 External Clock Check Target (OCCS_CLKCHKT)....341
18.5.8 Protection Register (OCCS_PROT)....341
18.6 Functional Description....342
18.7 Relaxation Oscillators....346
18.7.1 Trimming Frequency on the Internal 8 MHz Relaxation Oscillator....346
18.7.2 Trimming Frequency on the Internal 32 kHz Relaxation Oscillator....346
18.8 External Reference....347
18.9 Crystal Oscillator....347
18.9.1 Switching Clock Sources....347
18.10 Phase Locked Loop....349
18.10.1 PLL Recommended Range of Operation....349
18.10.2 PLL Lock Time Specification....349
18.10.2.1 Lock Time Definition....349
18.10.2.2 Parametric Influences on Reaction Time....349
18.11 PLL Frequency Lock Detector Block....350
18.12 Loss of Reference Clock Detector....350

18.13 Resets....351
18.13.1 Resets....351
18.14 Clocks....351
18.15 Interrupts....352

Chapter 19
Flash Memory Controller (FMC)

19.1 Introduction....353
19.1.1 Overview....353
19.1.2 Features....354
19.2 Modes of operation....354
19.3 External signal description....354
19.4 Memory map and register descriptions....355
19.4.1 Flash Access Protection Register (FMC_PFAPR)....359
19.4.2 Flash Bank 0 Control Register (FMC_PFB0CR)....361
19.4.3 Flash Bank 1 Control Register (FMC_PFB1CR)....363
19.4.4 Cache Tag Storage (FMC_TAGVDW0S*n*)....364
19.4.5 Cache Tag Storage (FMC_TAGVDW1S*n*)....365
19.4.6 Cache Tag Storage (FMC_TAGVDW2S*n*)....366
19.4.7 Cache Tag Storage (FMC_TAGVDW3S*n*)....367
19.4.8 Cache Data Storage (upper word) (FMC_DATAW0S*n*U)....367
19.4.9 Cache Data Storage (lower word) (FMC_DATAW0S*n*L)....368
19.4.10 Cache Data Storage (upper word) (FMC_DATAW1S*n*U)....368
19.4.11 Cache Data Storage (lower word) (FMC_DATAW1S*n*L)....369
19.4.12 Cache Data Storage (upper word) (FMC_DATAW2S*n*U)....369
19.4.13 Cache Data Storage (lower word) (FMC_DATAW2S*n*L)....370
19.4.14 Cache Data Storage (upper word) (FMC_DATAW3S*n*U)....370
19.4.15 Cache Data Storage (lower word) (FMC_DATAW3S*n*L)....371
19.5 Functional description....371
19.5.1 Default configuration....371

19.5.2 Configuration options..... 372
19.5.3 Wait states..... 372
19.5.4 Speculative reads..... 373
19.6 Initialization and application information..... 374

Chapter 20
Flash Memory Module (FTFL)

20.1 Introduction..... 375
20.1.1 Features..... 376
20.1.1.1 Program Flash Memory Features..... 376
20.1.1.2 FlexNVM Memory Features..... 376
20.1.1.3 FlexRAM Features..... 376
20.1.1.4 Other Flash Memory Module Features..... 377
20.1.2 Block Diagram..... 377
20.1.3 Glossary..... 378
20.2 External Signal Description..... 380
20.3 Memory Map and Registers..... 380
20.3.1 Flash Configuration Field Description..... 381
20.3.2 Program Flash IFR Map..... 381
20.3.2.1 Program Once Field..... 382
20.3.3 Data Flash IFR Map..... 382
20.3.3.1 EEPROM Data Set Size..... 382
20.3.3.2 FlexNVM Partition Code..... 383
20.3.4 Register Descriptions..... 384
20.3.4.1 Flash Status Register (FTFL_FSTAT)..... 386
20.3.4.2 Flash Configuration Register (FTFL_FCNFG)..... 387
20.3.4.3 Flash Security Register (FTFL_FSEC)..... 389
20.3.4.4 Flash Option Register (FTFL_FOPT)..... 390
20.3.4.5 Flash Common Command Object Registers (FTFL_FCCOB*n*)..... 391
20.3.4.6 Program Flash Protection Registers (FTFL_FPROT*n*)..... 392

20.3.4.7 EEPROM Protection Register (FTFL_FEPROT)....394
20.3.4.8 Data Flash Protection Register (FTFL_FDPROT)....395
20.4 Functional Description....396
20.4.1 Flash Protection....396
20.4.2 FlexNVM Description....398
20.4.2.1 FlexNVM Block Partitioning for FlexRAM....398
20.4.2.2 EEPROM User Perspective....399
20.4.2.3 EEPROM Implementation Overview....400
20.4.2.4 Write endurance to FlexRAM for EEPROM....401
20.4.3 Interrupts....402
20.4.4 Flash Operation in Low-Power Modes....403
20.4.4.1 Wait Mode....403
20.4.4.2 Stop Mode....403
20.4.5 Functional Modes of Operation....403
20.4.6 Flash Reads and Ignored Writes....403
20.4.7 Read While Write (RWW)....404
20.4.8 Flash Program and Erase....404
20.4.9 Flash Command Operations....404
20.4.9.1 Command Write Sequence....405
20.4.9.2 Flash Commands....407
20.4.9.3 Flash Commands by Mode....410
20.4.9.4 Allowed Simultaneous Flash Operations....410
20.4.10 Margin Read Commands....411
20.4.11 Flash Command Description....412
20.4.11.1 Read 1s Block Command....413
20.4.11.2 Read 1s Section Command....414
20.4.11.3 Program Check Command....415
20.4.11.4 Read Resource Command....416
20.4.11.5 Program Longword Command....417

20.4.11.6 Erase Flash Block Command....419
20.4.11.7 Erase Flash Sector Command....420
20.4.11.8 Program Section Command....423
20.4.11.9 Read 1s All Blocks Command....425
20.4.11.10 Read Once Command....426
20.4.11.11 Program Once Command....427
20.4.11.12 Erase All Blocks Command....428
20.4.11.13 Verify Backdoor Access Key Command....429
20.4.11.14 Program Partition Command....430
20.4.11.15 Set FlexRAM Function Command....432
20.4.12 Security....434
20.4.12.1 Flash Memory Access by Mode and Security....434
20.4.12.2 Changing the Security State....435
20.4.13 Reset Sequence....436

Chapter 21
Computer Operating Properly (COP) Watchdog

21.1 Introduction....437
21.1.1 Features....437
21.1.2 Block Diagram....437
21.2 Memory Map and Registers....438
21.2.1 COP Control Register (COP_CTRL)....439
21.2.2 COP Timeout Register (COP_TOUT)....441
21.2.3 COP Counter Register (COP_CNTR)....441
21.2.4 COP Interrupt Value Register (COP_INTVAL)....442
21.3 Functional Description....442
21.3.1 COP after Reset....443
21.3.2 Wait Mode Operation....443
21.3.3 Stop Mode Operation....443
21.3.4 Debug Mode Operation....443

21.3.5 Loss of Reference Operation....443
21.4 Resets....444
21.5 Clocks....444
21.6 Interrupts....444

Chapter 22
External Watchdog Monitor (EWM)

22.1 Introduction....445
22.1.1 Features....445
22.1.2 Modes of Operation....446
22.1.2.1 Stop Mode....446
22.1.2.2 Wait Mode....446
22.1.2.3 Debug Mode....447
22.1.3 Block Diagram....447
22.2 EWM Signal Descriptions....448
22.3 Memory Map/Register Definition....448
22.3.1 Control Register (EWM_CTRL)....448
22.3.2 Service Register (EWM_SERV)....450
22.3.3 Compare Low Register (EWM_CMPL)....450
22.3.4 Compare High Register (EWM_CMPH)....451
22.3.5 Clock Control Register (EWM_CLKCTRL)....451
22.3.6 Clock Prescaler Register (EWM_CLKPRESCALER)....452
22.4 Functional Description....453
22.4.1 The EWM_out Signal....453
22.4.2 The EWM_in Signal....454
22.4.3 EWM Counter....454
22.4.4 EWM Compare Registers....454
22.4.5 EWM Refresh Mechanism....455
22.4.6 EWM Interrupt....455
22.4.7 Selecting the EWM counter clock....456

22.4.8 Counter clock prescaler....456

Chapter 23
Cyclic Redundancy Check (CRC)

23.1 Introduction....457
23.1.1 Features....457
23.1.2 Block diagram....457
23.1.3 Modes of operation....458
23.1.3.1 Run mode....458
23.1.3.2 Low-power modes (Wait or Stop)....458
23.2 Memory map and register descriptions....458
23.2.1 CRC Data register (CRC_DATA)....459
23.2.2 CRC Polynomial register (CRC_GPOLY)....460
23.2.3 CRC Control register (CRC_CTRL)....461
23.3 Functional description....462
23.3.1 CRC initialization/reinitialization....462
23.3.2 CRC calculations....462
23.3.2.1 16-bit CRC....462
23.3.2.2 32-bit CRC....463
23.3.3 Transpose feature....463
23.3.3.1 Types of transpose....464
23.3.4 CRC result complement....465

Chapter 24
16-bit SAR Analog-to-Digital Converter (ADC16)

24.1 Introduction....467
24.1.1 Features....467
24.1.2 Block diagram....468
24.2 ADC signal descriptions....469
24.2.1 Analog Power (VDDA)....470
24.2.2 Analog Ground (VSSA)....470

24.2.3 Voltage Reference Select....470
24.2.4 Analog Channel Inputs (ADx)....471
24.3 Memory map and register definitions....471
24.3.1 ADC Status and Control Registers 1 (ADC*x*_SC1*n*)....472
24.3.2 ADC Configuration Register 1 (ADC*x*_CFG1)....475
24.3.3 ADC Configuration Register 2 (ADC*x*_CFG2)....477
24.3.4 ADC Data Result Register (ADC*x*_R*n*)....478
24.3.5 Compare Value Registers (ADC*x*_CV*n*)....479
24.3.6 Status and Control Register 2 (ADC*x*_SC2)....480
24.3.7 Status and Control Register 3 (ADC*x*_SC3)....482
24.3.8 ADC Offset Correction Register (ADC*x*_OFS)....484
24.3.9 ADC Plus-Side Gain Register (ADC*x*_PG)....484
24.3.10 ADC Plus-Side General Calibration Value Register (ADC*x*_CLPD)....485
24.3.11 ADC Plus-Side General Calibration Value Register (ADC*x*_CLPS)....485
24.3.12 ADC Plus-Side General Calibration Value Register (ADC*x*_CLP4)....486
24.3.13 ADC Plus-Side General Calibration Value Register (ADC*x*_CLP3)....486
24.3.14 ADC Plus-Side General Calibration Value Register (ADC*x*_CLP2)....487
24.3.15 ADC Plus-Side General Calibration Value Register (ADC*x*_CLP1)....487
24.3.16 ADC Plus-Side General Calibration Value Register (ADC*x*_CLP0)....488
24.4 Functional description....488
24.4.1 Clock select and divide control....489
24.4.2 Voltage reference selection....490
24.4.3 Hardware trigger and channel selects....490
24.4.4 Conversion control....491
24.4.4.1 Initiating conversions....491
24.4.4.2 Completing conversions....492
24.4.4.3 Aborting conversions....492
24.4.4.4 Power control....493
24.4.4.5 Sample time and total conversion time....493

24.4.4.6 Conversion time examples....496
24.4.4.7 Hardware average function....498
24.4.5 Automatic compare function....499
24.4.6 Calibration function....500
24.4.7 User-defined offset function....501
24.4.8 Temperature sensor....502
24.4.9 MCU wait mode operation....503
24.4.10 MCU Normal Stop mode operation....504
24.4.10.1 Normal Stop mode with 504
24.4.10.2 Normal Stop mode with ADACK enabled....504
24.4.11 MCU Low-Power Stop mode operation....505
24.5 Initialization information....505
24.5.1 ADC module initialization example....505
24.5.1.1 Initialization sequence....505
24.5.1.2 Pseudo-code example....506
24.6 Application information....507
24.6.1 External pins and routing....507
24.6.1.1 Analog supply pins....507
24.6.1.2 Analog voltage reference pins....508
24.6.1.3 Analog input pins....509
24.6.2 Sources of error....509
24.6.2.1 Sampling error....509
24.6.2.2 Pin leakage error....510
24.6.2.3 Noise-induced errors....510
24.6.2.4 Code width and quantization error....511
24.6.2.5 Linearity errors....512
24.6.2.6 Code jitter, non-monotonicity, and missing codes....512

Chapter 25
12-bit Cyclic Analog-to-Digital Converter (ADC12)

Section number	Title	Page
25.1	Introduction	515
25.1.1	Overview	515
25.1.2	Features	515
25.1.3	Block Diagram	516
25.2	Signal Descriptions	517
25.2.1	Overview	517
25.2.2	External Signal Descriptions	518
25.2.2.1	Analog Input Pins (ANA[0:7] and ANB[0:7])	518
25.2.2.2	Voltage Reference Pins (VREFH and VREFL)	518
25.3	Memory Map and Registers	519
25.3.1	ADC Control Register 1 (ADC*x*_CTRL1)	523
25.3.2	ADC Control Register 2 (ADC*x*_CTRL2)	527
25.3.3	ADC Zero Crossing Control 1 Register (ADC*x*_ZXCTRL1)	529
25.3.4	ADC Zero Crossing Control 2 Register (ADC*x*_ZXCTRL2)	531
25.3.5	ADC Channel List Register 1 (ADC*x*_CLIST1)	532
25.3.6	ADC Channel List Register 2 (ADC*x*_CLIST2)	534
25.3.7	ADC Channel List Register 3 (ADC*x*_CLIST3)	536
25.3.8	ADC Channel List Register 4 (ADC*x*_CLIST4)	537
25.3.9	ADC Sample Disable Register (ADC*x*_SDIS)	539
25.3.10	ADC Status Register (ADC*x*_STAT)	540
25.3.11	ADC Ready Register (ADC*x*_RDY)	542
25.3.12	ADC Low Limit Status Register (ADC*x*_LOLIMSTAT)	542
25.3.13	ADC High Limit Status Register (ADC*x*_HILIMSTAT)	543
25.3.14	ADC Zero Crossing Status Register (ADC*x*_ZXSTAT)	543
25.3.15	ADC Result Registers with sign extension (ADC*x*_RSLT*n*)	544
25.3.16	ADC Low Limit Registers (ADC*x*_LOLIM*n*)	545
25.3.17	ADC High Limit Registers (ADC*x*_HILIM*n*)	546
25.3.18	ADC Offset Registers (ADC*x*_OFFST*n*)	546
25.3.19	ADC Power Control Register (ADC*x*_PWR)	547

25.3.20 ADC Calibration Register (ADC*x*_CAL)..... 550
25.3.21 Gain Control 1 Register (ADC*x*_GC1)..... 551
25.3.22 Gain Control 2 Register (ADC*x*_GC2)..... 552
25.3.23 ADC Scan Control Register (ADC*x*_SCTRL)..... 554
25.3.24 ADC Power Control Register (ADC*x*_PWR2)..... 555
25.3.25 ADC Control Register 3 (ADC*x*_CTRL3)..... 556
25.3.26 ADC Scan Halted Interrupt Enable Register (ADC*x*_SCHLTEN)..... 557
25.4 Functional Description..... 557
25.4.1 Input Multiplex Function..... 560
25.4.2 ADC Sample Conversion Operating Modes..... 562
25.4.2.1 Normal Mode Operation..... 563
25.4.3 ADC Data Processing..... 565
25.4.4 Sequential Versus Parallel Sampling..... 566
25.4.5 Scan Sequencing..... 567
25.4.6 Enabling Additional Sample Slots for On-Chip Signals..... 568
25.4.7 Power Management..... 568
25.4.7.1 Low Power Modes..... 569
25.4.7.2 Startup in Different Power Modes..... 570
25.4.7.3 Stop Mode of Operation..... 571
25.5 Reset..... 571
25.6 Clocks..... 571
25.7 Interrupts..... 574
25.8 Timing Specifications..... 575

Chapter 26
Comparator (CMP)

26.1 Introduction..... 577
26.1.1 CMP features..... 577
26.1.2 6-bit DAC key features..... 578
26.1.3 ANMUX key features..... 578

26.1.4 CMP, DAC and ANMUX diagram....579
26.1.5 CMP block diagram....580
26.2 Memory map/register definitions....581
26.2.1 CMP Control Register 0 (CMP*x*_CR0)....582
26.2.2 CMP Control Register 1 (CMP*x*_CR1)....583
26.2.3 CMP Filter Period Register (CMP*x*_FPR)....584
26.2.4 CMP Status and Control Register (CMP*x*_SCR)....585
26.2.5 DAC Control Register (CMP*x*_DACCR)....586
26.2.6 MUX Control Register (CMP*x*_MUXCR)....587
26.3 Functional description....588
26.3.1 CMP functional modes....588
26.3.1.1 Disabled mode (# 1)....590
26.3.1.2 Continuous mode (#s 2A & 2B)....590
26.3.1.3 Sampled, Non-Filtered mode (#s 3A & 3B)....591
26.3.1.4 Sampled, Filtered mode (#s 4A & 4B)....592
26.3.1.5 Windowed mode (#s 5A & 5B)....594
26.3.1.6 Windowed/Resampled mode (# 6)....596
26.3.1.7 Windowed/Filtered mode (#7)....597
26.3.2 Power modes....597
26.3.2.1 Wait mode operation....597
26.3.2.2 Stop mode operation....598
26.3.3 Startup and operation....598
26.3.4 Low-pass filter....598
26.3.4.1 Enabling filter modes....599
26.3.4.2 Latency issues....599
26.4 CMP interrupts....601
26.5 Digital-to-analog converter....602
26.6 DAC functional description....602
26.6.1 Voltage reference source select....602

26.7 DAC resets....603
26.8 DAC clocks....603
26.9 DAC interrupts....603

Chapter 27
12-bit Digital-to-Analog Converter (DAC)

27.1 Introduction....605
27.1.1 Overview....605
27.1.2 Features....605
27.1.3 Block Diagram....606
27.2 Memory Map and Registers....606
27.2.1 Control Register 0 (DAC_CTRL0)....607
27.2.2 Buffered Data Register (DAC_DATAREG_FMT0)....609
27.2.3 Buffered Data Register (DAC_DATAREG_FMT1)....610
27.2.4 Step Size Register (DAC_STEPVAL_FMT0)....610
27.2.5 Step Size Register (DAC_STEPVAL_FMT1)....611
27.2.6 Minimum Value Register (DAC_MINVAL_FMT0)....611
27.2.7 Minimum Value Register (DAC_MINVAL_FMT1)....612
27.2.8 Maximum Value Register (DAC_MAXVAL_FMT0)....612
27.2.9 Maximum Value Register (DAC_MAXVAL_FMT1)....613
27.2.10 Status Register (DAC_STATUS)....613
27.2.11 Control Register 1 (DAC_CTRL1)....614
27.3 Functional Description....614
27.3.1 Conversion modes....614
27.3.1.1 Asynchronous conversion mode....614
27.3.1.2 Synchronous conversion mode....615
27.3.2 Operation Modes....615
27.3.2.1 Normal Mode....615
27.3.2.2 DMA Support Mode....615
27.3.2.3 Automatic Mode....616

27.3.3 DAC settling time ... 618
27.3.4 Waveform Programming Example ... 619
27.3.5 Sources of Waveform Distortion ... 619
27.3.5.1 Switching Glitches ... 619
27.3.5.2 Slew Effects ... 620
27.3.5.3 Clipping Effects (Automatic Mode Only) ... 620
27.4 Resets ... 620
27.5 Clocks ... 621
27.6 Interrupts ... 621

Chapter 28
Enhanced Flexible Pulse Width Modulator A (PWMA)

28.1 Introduction ... 623
28.1.1 Features ... 623
28.1.2 Modes of Operation ... 624
28.1.3 Block Diagram ... 624
28.1.3.1 PWM Submodule ... 625
28.2 Signal Descriptions ... 626
28.2.1 PWM[n]_A and PWM[n]_B - External PWM Output Pair ... 626
28.2.2 PWM[n]_X - Auxiliary PWM Output signal ... 626
28.2.3 FAULT[n] - Fault Inputs ... 627
28.2.4 PWM[n]_EXT_SYNC - External Synchronization Signal ... 627
28.2.5 EXT_FORCE - External Output Force Signal ... 627
28.2.6 PWM[n]_EXTA and PWM[n]_EXTB - Alternate PWM Control Signals ... 627
28.2.7 PWM[n]_OUT_TRIG0 and PWM[n]_OUT_TRIG1 - Output Triggers ... 627
28.2.8 EXT_CLK - External Clock Signal ... 628
28.3 Memory Map and Registers ... 628
28.3.1 Counter Register (PWMA_SM*n*CNT) ... 636
28.3.2 Initial Count Register (PWMA_SM*n*INIT) ... 637
28.3.3 Control 2 Register (PWMA_SM*n*CTRL2) ... 638

28.3.4 Control Register (PWMA_SM*n*CTRL)....640
28.3.5 Value Register 0 (PWMA_SM*n*VAL0)....642
28.3.6 Fractional Value Register 1 (PWMA_SM*n*FRACVAL1)....643
28.3.7 Value Register 1 (PWMA_SM*n*VAL1)....643
28.3.8 Fractional Value Register 2 (PWMA_SM*n*FRACVAL2)....644
28.3.9 Value Register 2 (PWMA_SM*n*VAL2)....644
28.3.10 Fractional Value Register 3 (PWMA_SM*n*FRACVAL3)....645
28.3.11 Value Register 3 (PWMA_SM*n*VAL3)....645
28.3.12 Fractional Value Register 4 (PWMA_SM*n*FRACVAL4)....646
28.3.13 Value Register 4 (PWMA_SM*n*VAL4)....646
28.3.14 Fractional Value Register 5 (PWMA_SM*n*FRACVAL5)....647
28.3.15 Value Register 5 (PWMA_SM*n*VAL5)....647
28.3.16 Fractional Control Register (PWMA_SM*n*FRCTRL)....648
28.3.17 Output Control Register (PWMA_SM*n*OCTRL)....649
28.3.18 Status Register (PWMA_SM*n*STS)....651
28.3.19 Interrupt Enable Register (PWMA_SM*n*INTEN)....653
28.3.20 DMA Enable Register (PWMA_SM*n*DMAEN)....654
28.3.21 Output Trigger Control Register (PWMA_SM*n*TCTRL)....656
28.3.22 Fault Disable Mapping Register 0 (PWMA_SM*n*DISMAP0)....657
28.3.23 Fault Disable Mapping Register 1 (PWMA_SM*n*DISMAP1)....657
28.3.24 Deadtime Count Register 0 (PWMA_SM*n*DTCNT0)....658
28.3.25 Deadtime Count Register 1 (PWMA_SM*n*DTCNT1)....659
28.3.26 Capture Control A Register (PWMA_SM*n*CAPTCTRLA)....659
28.3.27 Capture Compare A Register (PWMA_SM*n*CAPTCOMPA)....661
28.3.28 Capture Control B Register (PWMA_SM*n*CAPTCTRLB)....662
28.3.29 Capture Compare B Register (PWMA_SM*n*CAPTCOMPB)....663
28.3.30 Capture Control X Register (PWMA_SM*n*CAPTCTRLX)....664
28.3.31 Capture Compare X Register (PWMA_SM*n*CAPTCOMPX)....666
28.3.32 Capture Value 0 Register (PWMA_SM*n*CVAL0)....666

28.3.33 Capture Value 0 Cycle Register (PWMA_SM*n*CVAL0CYC)....666
28.3.34 Capture Value 1 Register (PWMA_SM*n*CVAL1)....667
28.3.35 Capture Value 1 Cycle Register (PWMA_SM*n*CVAL1CYC)....667
28.3.36 Capture Value 2 Register (PWMA_SM*n*CVAL2)....668
28.3.37 Capture Value 2 Cycle Register (PWMA_SM*n*CVAL2CYC)....668
28.3.38 Capture Value 3 Register (PWMA_SM*n*CVAL3)....668
28.3.39 Capture Value 3 Cycle Register (PWMA_SM*n*CVAL3CYC)....669
28.3.40 Capture Value 4 Register (PWMA_SM*n*CVAL4)....669
28.3.41 Capture Value 4 Cycle Register (PWMA_SM*n*CVAL4CYC)....670
28.3.42 Capture Value 5 Register (PWMA_SM*n*CVAL5)....670
28.3.43 Capture Value 5 Cycle Register (PWMA_SM*n*CVAL5CYC)....670
28.3.44 Output Enable Register (PWMA_OUTEN)....671
28.3.45 Mask Register (PWMA_MASK)....672
28.3.46 Software Controlled Output Register (PWMA_SWCOUT)....673
28.3.47 PWM Source Select Register (PWMA_DTSRCSEL)....674
28.3.48 Master Control Register (PWMA_MCTRL)....676
28.3.49 Master Control 2 Register (PWMA_MCTRL2)....677
28.3.50 Fault Control Register (PWMA_FCTRL*n*)....678
28.3.51 Fault Status Register (PWMA_FSTS*n*)....679
28.3.52 Fault Filter Register (PWMA_FFILT*n*)....680
28.3.53 Fault Test Register (PWMA_FTST*n*)....681
28.4 Functional Description....682
28.4.1 PWM Capabilities....682
28.4.1.1 Center Aligned PWMs....682
28.4.1.2 Edge Aligned PWMs....684
28.4.1.3 Phase Shifted PWMs....685
28.4.1.4 Double Switching PWMs....686
28.4.1.5 ADC Triggering....687
28.4.1.6 Enhanced Capture Capabilities (E-Capture)....689

28.4.1.7 Synchronous Switching of Multiple Outputs....691
28.4.2 Functional Details....691
28.4.2.1 PWM Clocking....692
28.4.2.2 Register Reload Logic....693
28.4.2.3 Counter Synchronization....693
28.4.2.4 PWM Generation....694
28.4.2.5 Output Compare Capabilities....696
28.4.2.6 Force Out Logic....696
28.4.2.7 Independent or Complementary Channel Operation....698
28.4.2.8 Deadtime Insertion Logic....698
28.4.2.9 Fractional Delay Logic....703
28.4.2.10 Output Logic....704
28.4.2.11 E-Capture....705
28.4.2.12 Fault Protection....706
28.4.3 PWM Generator Loading....711
28.4.3.1 Load Enable....711
28.4.3.2 Load Frequency....712
28.4.3.3 Reload Flag....713
28.4.3.4 Reload Errors....713
28.4.3.5 Initialization....714
28.5 Resets....714
28.6 Interrupts....715
28.7 DMA....716

Chapter 29
Programmable Delay Block (PDB)

29.1 Introduction....719
29.1.1 Features....719
29.1.2 Modes of Operation....720
29.1.3 Block Diagram....720

29.2 Memory Map and Registers....724
29.2.1 Master Control Register (PDB*x*_MCTRL)....725
29.2.2 Control A Register (PDB*x*_CTRLA)....727
29.2.3 Control C Register (PDB*x*_CTRLC)....729
29.2.4 DelayA Register (PDB*x*_DELAYA)....730
29.2.5 DelayB Register (PDB*x*_DELAYB)....731
29.2.6 DelayC Register (PDB*x*_DELAYC)....732
29.2.7 DelayD Register (PDB*x*_DELAYD)....732
29.2.8 Modulus Register (PDB*x*_MOD)....733
29.2.9 Counter Register (PDB*x*_CNTR)....734
29.3 Functional Description....734
29.3.1 Miscellaneous Concerns and SoC Integration....734
29.3.2 Impact of Using the Prescaler on Timing Resolution....735
29.3.3 Fault conditions....735
29.4 Resets....735
29.5 Clocks....735
29.6 Interrupts....735

Chapter 30
Quad Timer (TMR)

30.1 Overview....737
30.2 Features....738
30.3 Modes of Operation....738
30.4 Block Diagram....738
30.5 Memory Map and Registers....739
30.5.1 Timer Channel Compare Register 1 (TMR*x*_*n*COMP1)....743
30.5.2 Timer Channel Compare Register 2 (TMR*x*_*n*COMP2)....744
30.5.3 Timer Channel Capture Register (TMR*x*_*n*CAPT)....744
30.5.4 Timer Channel Load Register (TMR*x*_*n*LOAD)....744
30.5.5 Timer Channel Hold Register (TMR*x*_*n*HOLD)....745

30.5.6 Timer Channel Counter Register (TMR*x*_*n*CNTR)....745
30.5.7 Timer Channel Control Register (TMR*x*_*n*CTRL)....745
30.5.8 Timer Channel Status and Control Register (TMR*x*_*n*SCTRL)....748
30.5.9 Timer Channel Comparator Load Register 1 (TMR*x*_*n*CMPLD1)....749
30.5.10 Timer Channel Comparator Load Register 2 (TMR*x*_*n*CMPLD2)....750
30.5.11 Timer Channel Comparator Status and Control Register (TMR*x*_*n*CSCTRL)....750
30.5.12 Timer Channel Input Filter Register (TMR*x*_*n*FILT)....752
30.5.13 Timer Channel DMA Enable Register (TMR*x*_*n*DMA)....753
30.5.14 Timer Channel Enable Register (TMR*x*_*n*ENBL)....754
30.6 Functional Description....754
30.6.1 General....754
30.6.2 Usage of Compare Registers....755
30.6.3 Usage of Compare Load Registers....756
30.6.4 Usage of the Capture Register....757
30.6.5 Functional Modes....757
30.6.5.1 Stop Mode....757
30.6.5.2 Count Mode....758
30.6.5.3 Edge-Count Mode....759
30.6.5.4 Gated-Count Mode....759
30.6.5.5 Quadrature-Count Mode....760
30.6.5.6 Quadrature-Count Mode with Index Input....761
30.6.5.7 Signed-Count Mode....762
30.6.5.8 Triggered-Count Mode 1....762
30.6.5.9 Triggered-Count Mode 2....763
30.6.5.10 One-Shot Mode....764
30.6.5.11 Cascade-Count Mode....765
30.6.5.12 Pulse-Output Mode....766
30.6.5.13 Fixed-Frequency PWM Mode....768
30.6.5.14 Variable-Frequency PWM Mode....768

30.7 Resets....772
30.7.1 General....772
30.8 Clocks....772
30.8.1 General....772
30.9 Interrupts....772
30.9.1 General....772
30.9.2 Description of Interrupt Operation....773
30.9.2.1 Timer Compare Interrupts....773
30.9.2.2 Timer Overflow Interrupts....774
30.9.2.3 Timer Input Edge Interrupts....774
30.10 DMA....774

Chapter 31
Periodic Interrupt Timer (PIT)

31.1 Introduction....775
31.1.1 Features....775
31.1.2 Modes of Operation....775
31.1.3 Block Diagram....775
31.2 Memory Map and Registers....776
31.2.1 PIT Control Register (PIT*x*_CTRL)....777
31.2.2 PIT Modulo Register (PIT*x*_MOD)....778
31.2.3 PIT Counter Register (PIT*x*_CNTR)....779
31.3 Functional Description....779
31.3.1 Slave Mode....779
31.3.2 Low Power Modes....780
31.3.2.1 Wait Mode....780
31.3.2.2 Stop Mode....780
31.3.2.3 Debug Mode....781
31.4 Interrupts....781

Chapter 32

CAN (FlexCAN)

32.1 Introduction....783

32.1.1 Overview....784

32.1.2 FlexCAN module features....785

32.1.3 Modes of operation....786

32.2 FlexCAN signal descriptions....788

32.2.1 CAN Rx....788

32.2.2 CAN Tx....788

32.3 Memory map/register definition....789

32.3.1 FlexCAN memory mapping....789

32.3.2 Module Configuration Register (CAN_MCR)....792

32.3.3 Control 1 Register (CAN_CTRL1)....797

32.3.4 Free Running Timer (CAN_TIMER)....800

32.3.5 Rx Mailboxes Global Mask Register (CAN_RXMGMASK)....801

32.3.6 Rx 14 Mask Register (CAN_RX14MASK)....802

32.3.7 Rx 15 Mask Register (CAN_RX15MASK)....802

32.3.8 Error Counter (CAN_ECR)....803

32.3.9 Error and Status 1 Register (CAN_ESR1)....804

32.3.10 Interrupt Masks 2 Register (CAN_IMASK2)....809

32.3.11 Interrupt Masks 1 Register (CAN_IMASK1)....809

32.3.12 Interrupt Flags 2 Register (CAN_IFLAG2)....810

32.3.13 Interrupt Flags 1 Register (CAN_IFLAG1)....810

32.3.14 Control 2 Register (CAN_CTRL2)....813

32.3.15 Error and Status 2 Register (CAN_ESR2)....816

32.3.16 CRC Register (CAN_CRCR)....817

32.3.17 Rx FIFO Global Mask Register (CAN_RXFGMASK)....818

32.3.18 Rx FIFO Information Register (CAN_RXFIR)....819

32.3.19 Rx Individual Mask Registers (CAN_RXIMR*n*)....820

32.3.20 Message buffer structure....821

32.3.21 Rx FIFO structure ... 826
32.4 Functional description ... 828
32.4.1 Transmit process ... 829
32.4.2 Arbitration process ... 830
32.4.2.1 Lowest-number Mailbox first ... 830
32.4.2.2 Highest-priority Mailbox first ... 831
32.4.2.3 Arbitration process (continued) ... 832
32.4.3 Receive process ... 833
32.4.4 Matching process ... 835
32.4.5 Move process ... 840
32.4.5.1 Move-in ... 840
32.4.5.2 Move-out ... 841
32.4.6 Data coherence ... 842
32.4.6.1 Transmission abort mechanism ... 842
32.4.6.2 Mailbox inactivation ... 843
32.4.6.3 Mailbox lock mechanism ... 844
32.4.7 Rx FIFO ... 845
32.4.8 CAN protocol related features ... 847
32.4.8.1 Remote frames ... 847
32.4.8.2 Overload frames ... 848
32.4.8.3 Time stamp ... 848
32.4.8.4 Protocol timing ... 848
32.4.8.5 Arbitration and matching timing ... 851
32.4.9 Clock domains and restrictions ... 853
32.4.10 Modes of operation details ... 854
32.4.10.1 Freeze mode ... 854
32.4.10.2 Module Disable mode ... 855
32.4.10.3 Doze mode ... 856
32.4.10.4 Stop mode ... 857

32.4.11 Interrupts ... 859
32.4.12 Bus interface ... 860
32.5 Initialization/application information ... 861
32.5.1 FlexCAN initialization sequence ... 861

Chapter 33
Queued Serial Communications Interface (QSCI)

33.1 Introduction ... 863
33.1.1 Features ... 863
33.1.2 SCI Block Diagram ... 864
33.2 External Signal Descriptions ... 865
33.2.1 TXD —Transmit Data ... 865
33.2.2 RXD —Receiver Data ... 865
33.3 Memory Map and Registers ... 865
33.3.1 QSCI Baud Rate Register (QSCI*x*_RATE) ... 866
33.3.2 QSCI Control Register 1 (QSCI*x*_CTRL1) ... 866
33.3.3 QSCI Control Register 2 (QSCI*x*_CTRL2) ... 869
33.3.4 QSCI Status Register (QSCI*x*_STAT) ... 871
33.3.5 QSCI Data Register (QSCI*x*_DATA) ... 874
33.3.6 QSCI Control Register 3 (QSCI*x*_CTRL3) ... 875
33.4 Functional Description ... 876
33.4.1 Data Frame Format ... 876
33.4.2 Baud-Rate Generation ... 877
33.4.3 Transmitter ... 878
33.4.3.1 Character Length ... 879
33.4.3.2 Character Transmission ... 879
33.4.3.3 Break Characters ... 881
33.4.3.4 Preambles ... 881
33.4.4 Receiver ... 882
33.4.4.1 Character Length ... 882

33.4.4.2 Character Reception....883
33.4.4.3 Data Sampling....883
33.4.4.4 Framing Errors....888
33.4.4.5 Baud-Rate Tolerance....888
33.4.4.6 Slow Data Tolerance....888
33.4.4.7 Fast Data Tolerance....889
33.4.4.8 Receiver Wakeup....890
33.4.4.9 Single-Wire Operation....891
33.4.4.10 Loop Operation....892
33.4.5 DMA Operation....892
33.4.5.1 Transmit DMA Operation....892
33.4.5.2 Receive DMA Operation....893
33.4.5.3 Receiver Wakeup with DMA....893
33.4.6 LIN Slave Operation....893
33.4.7 Low-Power Options....894
33.4.7.1 Run Mode....894
33.4.7.2 Wait Mode....894
33.4.7.3 Stop Mode....894
33.5 Resets....895
33.6 Clocks....895
33.7 Interrupts....895
33.7.1 Description of Interrupt Operation....895
33.7.1.1 Transmitter Empty Interrupt....896
33.7.1.2 Transmitter Idle Interrupt....896
33.7.1.3 Receiver Full Interrupt....896
33.7.1.4 Receiver Edge Interrupt....897
33.7.1.5 Receive Error Interrupt....897
33.7.2 Recovery from Wait and Stop Mode....897
33.8 DMA Requests....897

33.8.1 Transmit Data Write Request....898
33.8.2 Receive Data Read Request....898

Chapter 34
Queued Serial Peripheral Interface (QSPI)

34.1 Introduction....899
34.1.1 Overview....899
34.1.2 Block Diagram....901
34.2 Signal Descriptions....902
34.2.1 External I/O Signals....902
34.2.1.1 MISO (Master In/Slave Out)....902
34.2.1.2 MOSI (Master Out/Slave In)....902
34.2.1.3 SCLK (Serial Clock)....902
34.2.1.4 SS (Slave Select)....903
34.3 Memory Map Registers....904
34.3.1 SPI Status and Control Register (QSPI_SPSCR)....904
34.3.2 SPI Data Size and Control Register (QSPI_SPDSR)....907
34.3.3 SPI Data Receive Register (QSPI_SPDRR)....910
34.3.4 SPI Data Transmit Register (QSPI_SPDTR)....911
34.3.5 SPI FIFO Control Register (QSPI_SPFIFO)....913
34.3.6 SPI Word Delay Register (QSPI_SPWAIT)....915
34.3.7 SPI Control Register 2 (QSPI_SPCTL2)....915
34.4 Functional Description....916
34.4.1 Operating Modes....916
34.4.1.1 Master Mode....916
34.4.1.2 Slave Mode....917
34.4.1.3 DMA Mode....918
34.4.1.4 Wired-OR Mode....919
34.4.2 Transaction Formats....919
34.4.2.1 Data Transaction Length....919

34.4.2.2 Data Shift Ordering....920
34.4.2.3 Clock Phase and Polarity Controls....920
34.4.2.4 Transaction Format When CPHA = 0....920
34.4.2.5 Transaction Format When CPHA = 1....922
34.4.2.6 Transaction Initiation Latency....923
34.4.2.7 SS Hardware-Generated Timing in Master Mode....923
34.4.3 Transmission Data....925
34.4.4 Error Conditions....926
34.4.4.1 Overflow Error....926
34.4.4.2 Mode Fault Error....928
34.4.5 Resetting the SPI....930
34.5 Interrupts....931

Chapter 35
Inter-Integrated Circuit (I2C)

35.1 Introduction....933
35.1.1 Features....933
35.1.2 Modes of operation....934
35.1.3 Block diagram....934
35.2 I2C signal descriptions....935
35.3 Memory map/register definition....936
35.3.1 I2C Address Register 1 (I2C*x*_A1)....937
35.3.2 I2C Frequency Divider register (I2C*x*_F)....937
35.3.3 I2C Control Register 1 (I2C*x*_C1)....938
35.3.4 I2C Status register (I2C*x*_S)....940
35.3.5 I2C Data I/O register (I2C*x*_D)....942
35.3.6 I2C Control Register 2 (I2C*x*_C2)....943
35.3.7 I2C Programmable Input Glitch Filter register (I2C*x*_FLT)....944
35.3.8 I2C Range Address register (I2C*x*_RA)....946
35.3.9 I2C SMBus Control and Status register (I2C*x*_SMB)....946

35.3.10 I2C Address Register 2 (I2C*x*_A2)....948
35.3.11 I2C SCL Low Timeout Register High (I2C*x*_SLTH)....948
35.3.12 I2C SCL Low Timeout Register Low (I2C*x*_SLTL)....949
35.4 Functional description....949
35.4.1 I2C protocol....949
35.4.1.1 START signal....950
35.4.1.2 Slave address transmission....950
35.4.1.3 Data transfers....951
35.4.1.4 STOP signal....951
35.4.1.5 Repeated START signal....952
35.4.1.6 Arbitration procedure....952
35.4.1.7 Clock synchronization....952
35.4.1.8 Handshaking....953
35.4.1.9 Clock stretching....953
35.4.1.10 I2C divider and hold values....953
35.4.2 10-bit address....954
35.4.2.1 Master-transmitter addresses a slave-receiver....955
35.4.2.2 Master-receiver addresses a slave-transmitter....955
35.4.3 Address matching....956
35.4.4 System management bus specification....957
35.4.4.1 Timeouts....957
35.4.4.2 FAST ACK and NACK....959
35.4.5 Resets....959
35.4.6 Interrupts....959
35.4.6.1 Byte transfer interrupt....960
35.4.6.2 Address detect interrupt....960
35.4.6.3 Stop Detect Interrupt....961
35.4.6.4 Exit from low-power/stop modes....961
35.4.6.5 Arbitration lost interrupt....961

35.4.6.6 Timeout interrupt in SMBus ... 961
35.4.7 Programmable input glitch filter ... 962
35.4.8 Address matching wake-up ... 962
35.4.9 DMA support ... 963
35.5 Initialization/application information ... 963

Chapter 36
General-Purpose Input/Output (GPIO)

36.1 Overview ... 967
36.1.1 Features ... 967
36.1.2 Modes of Operation ... 968
36.2 Memory Map and Registers ... 968
36.2.1 GPIO Pull Resistor Enable Register (GPIO*x*_PUR) ... 971
36.2.2 GPIO Data Register (GPIO*x*_DR) ... 972
36.2.3 GPIO Data Direction Register (GPIO*x*_DDR) ... 973
36.2.4 GPIO Peripheral Enable Register (GPIO*x*_PER) ... 973
36.2.5 GPIO Interrupt Assert Register (GPIO*x*_IAR) ... 974
36.2.6 GPIO Interrupt Enable Register (GPIO*x*_IENR) ... 974
36.2.7 GPIO Interrupt Polarity Register (GPIO*x*_IPOLR) ... 975
36.2.8 GPIO Interrupt Pending Register (GPIO*x*_IPR) ... 975
36.2.9 GPIO Interrupt Edge Sensitive Register (GPIO*x*_IESR) ... 976
36.2.10 GPIO Push-Pull Mode Register (GPIO*x*_PPMODE) ... 976
36.2.11 GPIO Raw Data Register (GPIO*x*_RAWDATA) ... 977
36.2.12 GPIO Drive Strength Control Register (GPIO*x*_DRIVE) ... 978
36.2.13 GPIO Pull Resistor Type Select (GPIO*x*_PUS) ... 978
36.2.14 Slew Rate Control Register (GPIO*x*_SRE) ... 979
36.3 Functional Description ... 979
36.4 Interrupts ... 980
36.5 Clocks and Resets ... 981

Chapter 1
About This Document

1.1 Overview

1.1.1 Purpose

This document describes the features, architecture, and programming model of NXP's MWCT10x3A series of microcontrollers.

1.1.2 Audience

This document is primarily for system architects and software application developers who are using or considering using this chip in a system.

1.2 Conventions

1.2.1 Numbering systems

The following suffixes identify different numbering systems:

This suffix	Identifies a
b	Binary number. For example, the binary equivalent of the number 5 is written 101b. In some cases, binary numbers are shown with the prefix *0b*.
d	Decimal number. Decimal numbers are followed by this suffix only when the possibility of confusion exists. In general, decimal numbers are shown without a suffix.
h	Hexadecimal number. For example, the hexadecimal equivalent of the number 60 is written 3Ch. In some cases, hexadecimal numbers are shown with the prefix *0x*.

1.2.2 Typographic notation

The following typographic notation is used throughout this document:

Example	Description
placeholder, x	Items in italics are placeholders for information that you provide. Italicized text is also used for the titles of publications and for emphasis. Plain lowercase letters are also used as placeholders for single letters and numbers.
`code`	Fixed-width type indicates text that must be typed exactly as shown. It is used for instruction mnemonics, directives, symbols, subcommands, parameters, and operators. Fixed-width type is also used for example code. Instruction mnemonics and directives in text and tables are shown in all caps; for example, BSR.
SR[SCM]	A mnemonic in brackets represents a named field in a register. This example refers to the Scaling Mode (SCM) field in the Status Register (SR).
REVNO[6:4], XAD[7:0]	Numbers in brackets and separated by a colon represent either: • A subset of a register's named field For example, REVNO[6:4] refers to bits 6–4 that are part of the COREREV field that occupies bits 6–0 of the REVNO register. • A continuous range of individual signals of a bus For example, XAD[7:0] refers to signals 7–0 of the XAD bus.

1.2.3 Special terms

The following terms have special meanings:

Term	Meaning
asserted	Refers to the state of a signal as follows: • An active-high signal is asserted when high (1). • An active-low signal is asserted when low (0).
deasserted	Refers to the state of a signal as follows: • An active-high signal is deasserted when low (0). • An active-low signal is deasserted when high (1). In some cases, deasserted signals are described as *negated*.
reserved	Refers to a memory space, register, or field that is either reserved for future use or for which, when written to, the module or chip behavior is unpredictable.

Chapter 2
Introduction

2.1 Introduction

The MWCT10x3A microcontroller is a member of the 32-bit 56800EX core devices. It combines, on a single chip, the processing power of a 32-bit DSP and the functionality of a microcontroller with a flexible set of peripherals. Due to its cost-effectiveness, configuration flexibility, and compact program code, MWCT10x3A is well-suited for many consumer and automotive wireless charging.

The 56800EX core is based on a dual Harvard-style architecture, consisting of 3 execution units operating in parallel, allowing as many as 6 operations per instruction cycle. The MCU-style programming model and optimized instruction set allow straightforward generation of efficient, compact DSP and control code. The instruction set is also highly efficient for C compilers, to enable rapid development of optimized control applications. Additionally, memory resource protection (MRP) is provided to protect supervisor programs and resources from user programs.

The on-chip flash memory and RAM can be mapped into both the program and data memory spaces. Two data operands can be accessed from the on-chip data RAM per instruction cycle.

2.1.1 Core Overview

The MWCT10x3A is based on an 56800EX core, which updates the 56800E core. The 56800EX core has all 56800E core features and adds new enhancements, including:

- 32-bit x 32-bit MUL/MAC operations
- all registers in the Address Generation Unit (AGU) have shadowed registers that effectively reduce the context save/restore time during exception processing, thereby reducing latency.

- bit-reverse address mode supporting Fast Fourier Transform
- new bit manipulation instruction that integrates a Test bitfield and a Set/Clear (BFSC) bitfield into a single instruction

With all existing 32-bit arithmetic operations, the 56800EX core is truly 32-bit compatible.

2.1.2 Memory Overview

MWCT10x3A includes multiple blocks of on-chip memory

- Up to 256 KB (128 KW) program flash memory
- 32 KB (16 KW) FlevNVM/FlexRAM
- 288 KB (144 KW) Total flash memory, including FlexNVM
- Up to 32 KB (16 KW) RAM

Both bulk erasing and erasing in pages are supported.

2.1.3 Peripheral Overview

A full set of programmable peripherals—including PWM, ADC, QSCIs, QSPI, I2C, a FlexCAN module, Inter-Module Crossbars, Quad Timers, a CRC block, 12-bit DAC, Analog Comparators, and on-chip/off-chip clock sources—supports various applications. Each peripheral's clock can be independently gated to save power. Any pin in these peripherals can also be used as General Purpose Input/Outputs (GPIOs).

2.2 Application Examples

With numerous, highly integrated peripherals and powerful processing capabilities, MWCT10x3A is a low-cost family especially useful for wireless charging applications.

2.3 Features

The following list summarizes the superset of features across the entire MWCT10x3A family.

- 56800EX 32-bit core
- Up to 100 MHz operation frequency
- Up to 128 KW program flash memory

- Up to 16 KW dual port program/data RAM
- Memory resource protection (MRP) unit:
  - Partitions software into two modes—supervisor software and user software—with separate system address spaces and resources, for both program and data
  - Protects supervisor programs and resources from user programs
- Four-channel DMA
- One 8-channel eFlexPWM module with NanoEdge™ placement and enhanced capture
- One -channel eFlexPWM module
- 2 x 8-channel 12-bit cyclic ADC
- 1 x 24-channel 16-bit SAR ADC with temperature sensor
- One windowed watchdog timer
- Cyclic Redundancy Check (CRC) generator
- On-chip 8 MHz/400 kHz relaxation oscillator, 32 kHz Relaxation Oscillator and 4 MHz to 16 MHz Crystal Oscillator (XOSC)
- Power Supervisor
- Inter-Module Crossbar with AND-OR-INVERT function
- Programmable Interrupt Controller (INTC)
- Two Quad Timer
- Two Periodic Interval Timers
- One 12-bit DAC module
- Four 6-bit DACs (64-tap voltage reference)
- Four High Speed Comparators
- One Queued SPI modules
- Two Queued SCI modules
- Two I2C/SMBus module
- One FlexCAN module
- 5 V tolerant I/O (except for RESETB pin which is a 3.3 V pin)

2.3.1 MWCT10x3A product features

Table 2-1. MWCT10x3AVLH product features

Core frequency (MHz)	100
Flash memory (KB)	256
FlevNVM/ FlexRAM (KB)	32/2
Total flash memory, including FlexNVM (KB)	288
RAM (KB)	32
Memory resource protection	Yes
External Watchdog	1
12-bit Cyclic ADC channels	2x8 (300 ns)

Table continues on the next page...

Table 2-1. MWCT10x3AVLH product features (continued)

16-bit SAR ADC (with Temp Sensor) channels	1x8
PWMA with input capture: High resolution channels Standard channels	 1x8 1
12-bit DAC	1
DMA	Yes
CMP + 6-bit DAC	4
QSCI	2
QSPI	1
I2C/SMBus	2
FlexCAN	1
LQFP package pin count	64

2.3.2 Block Diagram

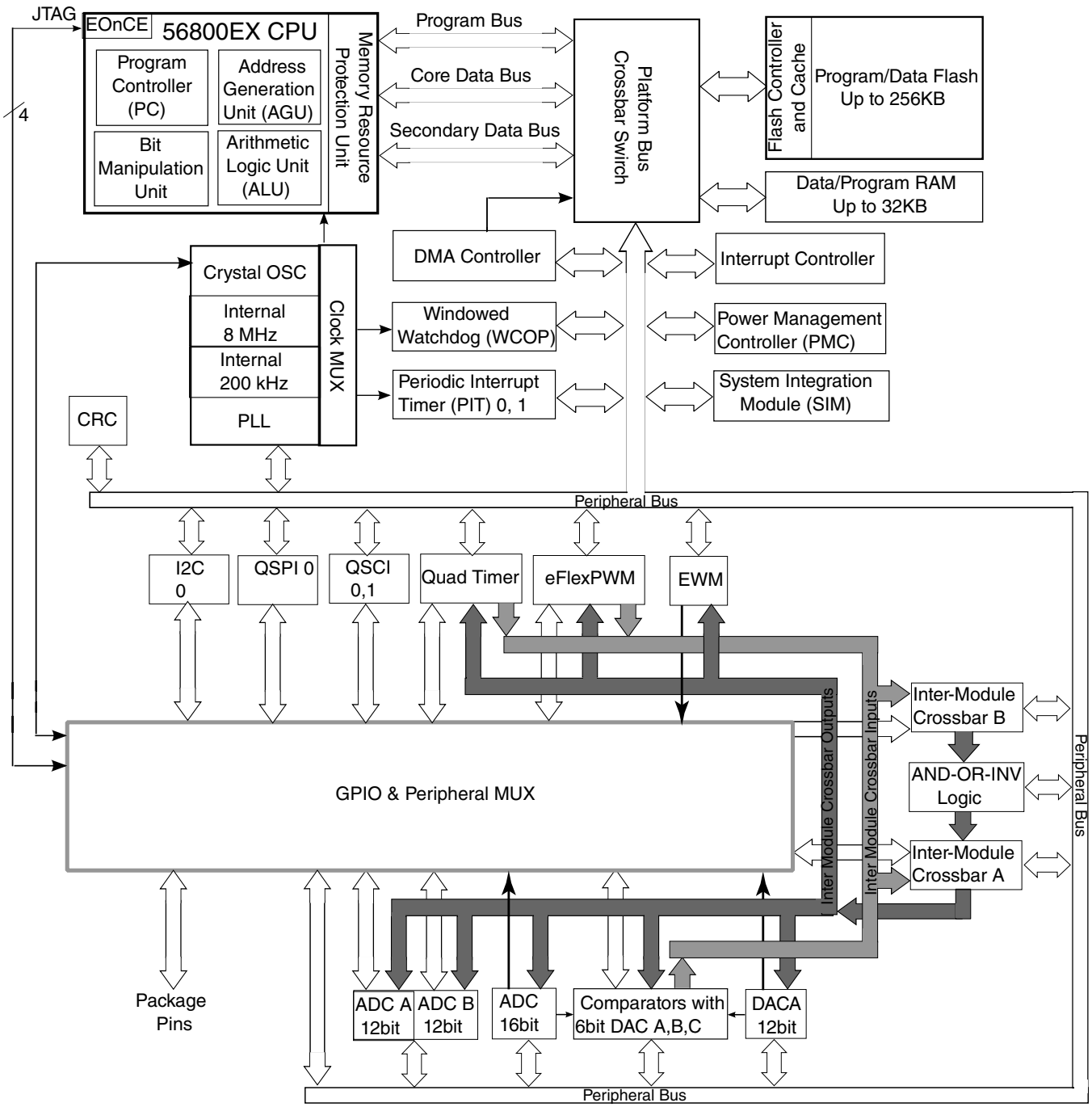


Figure 2-1. Block Diagram

2.3.3 56800EX 32-bit core

- Efficient 32-bit 56800EX Digital Signal Processor (DSP) engine with modified dual Harvard architecture:
  - Three internal address buses
  - Four internal data buses: two 32-bit primary buses, one 16-bit secondary data bus, and one 16-bit instruction bus
  - 32-bit data accesses
  - Supports concurrent instruction fetches in the same cycle, and dual data accesses in the same cycle
  - 20 addressing modes
- As many as 100 million instructions per second (MIPS) at 100 MHz core frequency
- 162 basic instructions
- Instruction set supports both fractional arithmetic and integer arithmetic
- 32-bit internal primary data buses support 8-bit, 16-bit, and 32-bit data movement, plus addition, subtraction, and logical operations
- Single-cycle 16 × 16-bit -> 32-bit and 32 x 32-bit -> 64-bit multiplier-accumulator (MAC) with dual parallel moves
- 32-bit arithmetic and logic multi-bit shifter
- Four 36-bit accumulators, including extension bits
- Parallel instruction set with unique DSP addressing modes
- Hardware DO and REP loops
- Bit reverse address mode, which effectively supports DSP and Fast Fourier Transform algorithms
- Full shadowing of the register stack for zero-overhead context saves and restores: nine shadow registers correspond to nine address registers (R0, R1, R2, R3, R4, R5, N, N3, M01)
- Instruction set supports both DSP and controller functions
- Controller-style addressing modes and instructions enable compact code
- Enhanced bit manipulation instruction set
- Efficient C compiler and local variable support
- Software subroutine and interrupt stack, with the stack's depth limited only by memory
- Priority level setting for interrupt levels
- JTAG/Enhanced On-Chip Emulation (OnCE) for unobtrusive, real-time debugging that is independent of processor speed

2.3.4 Operation Parameters

- Up to 100 MHz operation at -40 ºC to 105 ºC ambient temperature

- Single 3.3 V power supply
- Supply range: V_{DD} - V_{SS} = 2.7 V to 3.6 V, V_{DDA} - V_{SSA} = 2.7 V to 3.6 V

2.3.5 Packages

- 64LQFP

2.3.6 On-Chip Memory and Memory Protection

- Dual Harvard architecture permits as many as three simultaneous accesses to program and data memory
- Internal flash memory with security and protection to prevent unauthorized access
- Memory resource protection (MRP) unit to protect supervisor programs and resources from user programs
- Programming code can reside in flash memory during flash programming
- The dual-port RAM controller supports concurrent instruction fetches and data accesses, or dual data accesses by the core.
  - Concurrent accesses provide increased performance.
  - The data and instruction arrive at the core in the same cycle, reducing latency.
- On-chip memory
  - Up to 288 KB program/data flash memory, including FlexNVM
  - Up to 32 KB dual port data/program RAM

2.3.7 Peripherals

2.3.7.1 System Modules

2.3.7.1.1 Interrupt Controller

- Five interrupt priority levels
  - Three user-programmable priority levels for each interrupt source: level 0, level 1, level 2
  - Unmaskable level 3 interrupts include illegal instruction, hardware stack overflow, misaligned data access, SWI3 instruction
  - Interrupt level 3 is highest priority and non-maskable. Its sources include:
    - Illegal instructions
    - Hardware stack overflow
    - SWI instruction

        - EOnce interrupts
        - Misaligned data accesses
    - Lowest-priority software interrupt: level LP
- Support for nested interrupts, so that a higher priority level interrupt request can interrupt lower priority interrupt subroutine
- Masking of interrupt priority level is managed by the 56800EX core
- Two programmable fast interrupts that can be assigned to any interrupt source
- Notification to System Integration Module (SIM) to restart clock when in wait and stop states
- Ability to relocate interrupt vector table

2.3.7.1.2 Direct Memory Access (DMA) Controller

- Four independently programmable DMA controller channels
- Dual-address transfers via 32-bit master connection to the system bus
- Data transfers in 8-bit, 16-bit, or 32-bit blocks
- Continuous-mode or cycle-steal transfers from software or peripheral initiation
- One programmable input selected from 16 possible peripheral requests per channel
- Automatic hardware acknowledge/done indicator from each channel
- Independent source and destination address registers
- Optional modulo addressing and automatic updates of source and destination addresses
- Independent transfer sizes for source and destination
- Optional auto-alignment feature for source or destination accesses
- Optional automatic single or double channel linking
- Programming model accessed via 32-bit slave peripheral bus
- Channel arbitration on transfer boundaries using fixed priority scheme
- DMA peripherals:
    - Quad Timer
    - ADCs
    - QSPIs
    - QSCIs
    - I2Cs
    - PWMs
    - Crossbar
    - 12-bit DAC

2.3.7.1.3 Inter-Module Crossbar and AND-OR-INVERT logic

- Provides generalized connections between and among on-chip peripherals: ADCs, 12-bit DAC, comparators, quad-timers, eFlexPWMs, PDBs, EWM, and select I/O pins
- User-defined input/output pins for all modules connected to the crossbar

- DMA request and interrupt generation from the crossbar
- Write-once protection for all registers
- AND-OR-INVERT function provides a universal Boolean function generator that uses a four-term sum-of-products expression, with each product term containing true or complement values of the four selected inputs (A, B, C, D).

2.3.7.1.4 Cyclic Redundancy Check (CRC) Generator

- Hardware CRC generator circuit with 16-bit shift register
- High-speed hardware CRC calculation
- Programmable initial seed value
- CRC16-CCITT compliancy with $x^{16} + x^{12} + x^5 + 1$ polynomial
- Error detection for all single, double, odd, and most multibit errors
- Option to transpose input data or output data (CRC result) bitwise, which is required for certain CRC standards

2.3.7.2 General Purpose I/O (GPIO)

- 5 V tolerance (except RESETB pin)
- Individual control of peripheral mode or GPIO mode for each pin
- Programmable push-pull or open drain output
- Configurable pullup or pulldown on all input pins
- All pins (except JTAG and RESETB) default to be GPIO inputs
- 2 mA / 9 mA source/sink capability
- Controllable output slew rate

2.3.7.3 Timers and PWM modules

2.3.7.3.1 Enhanced Flex Pulse Width Modulator (eFlexPWM)

- Up to 100 MHz operation clock with PWM Resolution as fine as 10 ns
- PWM module contains four identical submodules, with up to three outputs per submodule
- PWM module contains four identical submodules, with two outputs per submodule
- 16 bits of resolution for center, edge-aligned, and asymmetrical PWMs
- PWMA with NanoEdge high resolution
  - Fractional delay for enhanced resolution of the PWM period and edge placement
  - Arbitrary PWM edge placement
  - 312 ps PWM frequency and duty-cycle resolution when NanoEdge functionality is enabled.

- PWM outputs can be configured as complementary output pairs or independent outputs
- Dedicated time-base counter with period and frequency control per submodule
- Independent top and bottom deadtime insertion for each complementary pair
- Independent control of both edges of each PWM output
- Enhanced input capture and output compare functionality on each input:
  - Channels not used for PWM generation can be used for buffered output compare functions.
  - Channels not used for PWM generation can be used for input capture functions.
  - Enhanced dual edge capture functionality
- Synchronization of submodule to external hardware (or other PWM) is supported.
- Double-buffered PWM registers
  - Integral reload rates from 1 to 16
  - Half-cycle reload capability
- Multiple output trigger events can be generated per PWM cycle via hardware.
- Support for double-switching PWM outputs
- Up to eight fault inputs can be assigned to control multiple PWM outputs
  - Programmable filters for fault inputs
- Independently programmable PWM output polarity
- Individual software control of each PWM output
- All outputs can be programmed to change simultaneously via a FORCE_OUT event.
- PWMX pin can optionally output a third PWM signal from each submodule
- Option to supply the source for each complementary PWM signal pair from any of the following:
  - Crossbar module outputs
  - External ADC input, taking into account values set in ADC high and low limit registers

2.3.7.3.2 Quad Timer

- Four 16-bit up/down counters, with a programmable prescaler for each counter
- Operation modes: edge count, gated count, signed count, capture, compare, PWM, signal shot, single pulse, pulse string, cascaded, quadrature decode
- Programmable input filter
- Counting start can be synchronized across counters

2.3.7.3.3 Periodic Interrupt Timer (PIT) Modules

- 16-bit up-counter with programmable counter modulo
- Interrupt capability
- Selectable clock sources:
  - External crystal oscillator/external clock source
  - On-chip low-power 32 kHz oscillator

    - System bus (IPBus up to 100 MHz)
    - 8 MHz / 400 kHz ROSC
- Can signal the device to exit powerdown mode
- Programmable master/slave selection between PIT instances

2.3.7.3.4 Computer Operating Properly (COP) Watchdog

- Programmable timeout period
- Support for operation in all power modes: run mode, wait mode, stop mode
- Causes loss of reference reset 128 cycles after loss of reference clock to the PLL is detected
- Selectable reference clock source in support of EN60730 and IEC61508
- Selectable clock sources:
    - External crystal oscillator/external clock source
    - On-chip low-power 32 kHz oscillator
    - System bus
    - 8 MHz / 400 kHz ROSC
- Support for interrupt triggered when the counter reaches the timeout value

2.3.7.3.5 External Watchdog Monitor (EWM)

- Monitors external circuit as well as the software flow
- Programmable timeout period
- Interrupt capability prior to timeout
- Independent output (EWM_OUT_b) that places external circuit (but not CPU and peripheral) in a safe mode when EWM timeout occurs
- Selectable reference clock source in support of EN60730 and IEC61508
- Wait mode and Stop mode operation is not supported.
- Selectable clock sources:
    - External crystal oscillator/external clock source
    - On-chip low-power 32 kHz oscillator
    - System bus (IPBus up to 100 MHz)
    - 8 MHz / 400 kHz ROSC

2.3.7.4 Clock Modules

2.3.7.4.1 On-chip oscillators

- Tunable 8 MHz relaxation oscillator with 400 kHz at standby mode (divide-by-two output)
- 32 kHz low frequency clock as secondary clock source for COP, EWM, PIT

2.3.7.4.2 Phase-locked loop

- Wide programmable output frequency: 240 MHz to 400 MHz
- Input reference clock frequency: 8 MHz to 16 MHz
- Detection of loss of lock and loss of reference clock
- Ability to power down

2.3.7.5 Analog Modules

2.3.7.5.1 12-bit Analog-to-Digital Converter (Cyclic type)

- Two independent 12-bit analog-to-digital converters (ADCs):
  - 2 x 8-channel external inputs
  - Built-in x1, x2, x4 programmable gain pre-amplifier
  - Maximum ADC clock frequency up to 20 MHz, having period as low as a 50-ns
  - Single conversion time of 8.5 ADC clock cycles
  - Additional conversion time of 6 ADC clock cycles
- Support of analog inputs for single-ended and differential conversions
- Sequential, parallel, and independent scan mode
- First 8 samples have offset, limit and zero-crossing calculation supported
- ADC conversions can be synchronized by *any* module connected to the internal crossbar module, such as PWM, timer, GPIO, and comparator modules.
- Support for simultaneous triggering and software-triggering conversions
- Support for a multi-triggering mode with a programmable number of conversions on each trigger
- Each ADC has ability to scan and store up to 8 conversion results.
- Current injection protection

2.3.7.5.2 16-bit Analog-to-Digital Converter (SAR type)

- Linear successive approximation algorithm with up to 16-bit resolution
- Differential and 16 single-ended external analog inputs
- Maximum ADC clock frequency up to 12.5 MHz
- Output modes: single-ended 16-bit, 12-bit, 10-bit, and 8-bit modes
- Output formatted in 2's complement, 16-bit sign extended for differential modes
- Output in right-justified, unsigned format for single-ended modes
- Single or continuous conversion (automatic return to idle after single conversion)
- Configurable sample time and conversion speed/power
- Conversion complete / hardware average complete flag and interrupt
- Input clock selectable from up to four sources
- Operation in low-power modes for lower-noise operation
- Asynchronous clock source for lower-noise operation with option to output the clock

- Automatic compare with interrupt for less-than, greater-than, or equal-to, within range or out-of-range programmable value
- Integrated temperature sensor
- Selectable voltage reference: internal, external, or alternate

2.3.7.5.3 12-bit Digital-to-Analog Converter

- 12-bit resolution
- Powerdown mode
- Automatic mode allows the DAC to automatically generate pre-programmed output waveforms, including square, triangle, and sawtooth waveforms (for applications like slope compensation)
- Programmable period, update rate, and range
- Output can be routed to an internal comparator, ADC, or optionally to an off-chip destination

2.3.7.5.4 6-bit Digital-to-Analog Converter

- 2.7 V to 3.3 V operation range
- 64-tap resistor ladder
- Selectable supply reference source
- Powerdown mode to conserve power when not in use
- Output routed to internal comparator input
- Less than 20 µA power consumption

2.3.7.5.5 Comparator

- Full rail-to-rail comparison range
- Support for high and low speed modes
- Selectable input source includes external pins and internal DACs
- Programmable output polarity
- 6-bit programmable DAC as a voltage reference per comparator
- Three programmable hysteresis levels
- Selectable interrupt on rising-edge, falling-edge, or toggle of a comparator output

2.3.7.6 Communication Interfaces

2.3.7.6.1 Queued Serial Peripheral Interface (QSPI) modules

- Maximum 25 Mbit/s baud rate
- Selectable baud rate clock sources for low baud rate communication
- Baud rate as low as Baudrate_Freq_in / 8192
- Full-duplex operation

- Master and slave modes
- Double-buffered operation with separate transmit and receive registers
- Four-word-deep FIFOs available on transmit and receive buffers
- Programmable length transmissions (2 bits to 16 bits)
- Programmable transmit and receive shift order (MSB as first bit transmitted)

2.3.7.6.2 Queued Serial Communications Interface (QSCI) modules

- Operating clock can be up to two times the CPU operating frequency
- Four-word-deep FIFOs available on both transmit and receive buffers
- Standard mark/space non-return-to-zero (NRZ) format
- 13-bit integer and 3-bit fractional baud rate selection
- Full-duplex or single-wire operation
- Programmable 8-bit or 9-bit data format
- Error detection capability
- Two receiver wakeup methods:
  - Idle line
  - Address mark
- 1/16 bit-time noise detection

2.3.7.6.3 Inter-Integrated Circuit (I2C)/System Management Bus (SMBus) modules

- Compatible with I2C bus standard
- Support for System Management Bus (SMBus) specification, version 2
- Multi-master operation
- General call recognition
- 10-bit address extension
- Start/Repeat and Stop indication flags
- Support for dual slave addresses or configuration of a range of slave addresses
- Programmable glitch input filter

2.3.7.6.4 Flex Controller Area Network (FlexCAN) module

- Clock source from PLL or XOSC/CLKIN
- Implementation of CAN protocol Version 2.0 A/B
- Standard and extended data frames
- Data length of 0 to 8 bytes
- Programmable bit rate up to 1 Mbps
- Support for remote frames
- Sixteen Message Buffers: each Message Buffer can be configured as receive or transmit, and supports standard and extended messages
- Individual Rx Mask Registers per Message Buffer

- Internal timer for time-stamping of received and transmitted messages
- Listen-only mode capability
- Programmable loopback mode, supporting self-test operation
- Programmable transmission priority scheme: lowest ID, lowest buffer number, or highest priority
- Global network time, synchronized by a specific message
- Low power modes, with programmable wakeup on bus activity

2.3.7.7 Power Management

2.3.7.7.1 On-Chip Voltage Regulator

- Input 2.7 V to 3.6 V (4.0 V absolute maximum rating)
- Provides 1.2 V ± 10% accuracy
- Separate large and small regulators
- Distributed type layout

2.3.7.7.2 Power supervisor

- Power-on reset (POR) to reset CPU, peripherals, and JTAG/EOnCE controllers (V_{DD} > 2.1 V)
- Brownout reset (V_{DD} < 1.9 V)
- Critical warn low-voltage interrupt (LVI2.0)
- Peripheral low-voltage interrupt (LVI2.7)

Chapter 3
Chip Configuration

3.1 Introduction

This chapter provides details on the individual modules of the microcontroller. It includes:

- module block diagrams showing immediate connections within the device,
- specific module-to-module interactions not necessarily discussed in the individual module chapters, and
- links for more information.

3.2 Core Configuration

This section summarizes how the module has been configured in the chip. For a comprehensive description of the module itself, see the module's dedicated chapter.

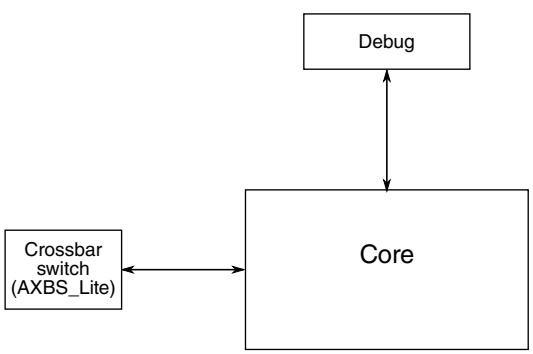


Figure 3-1. Core configuration

Table 3-1. Reference links to related information

Topic	Related module	Reference
Full description	Core	DSP56800E and DSP56800EX Cores Reference Manual (document ID: DSP56800ERM)

Table continues on the next page...

Table 3-1. Reference links to related information (continued)

Topic	Related module	Reference
System memory map		System memory map
Clocking		Clock Distribution
Power management		Power management

3.3 System modules

3.3.1 System Integration Module (SIM) Configuration

This section summarizes how the module has been configured in the chip. For a comprehensive description of the module itself, see the module's dedicated chapter.

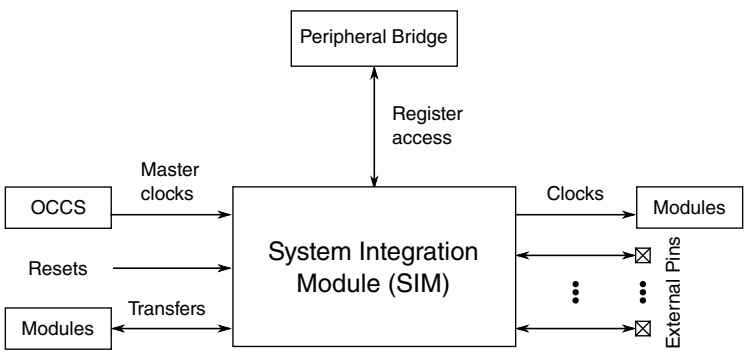


Figure 3-2. SIM configuration

Table 3-2. Reference links to related information

Topic	Related module	Reference
Full description	System Integration Module (SIM)	SIM
System memory map		System memory map
Clocking		Clock Distribution
Power management		Power management

3.3.2 MCM Configuration

This section summarizes how the module has been configured in the chip. For a comprehensive description of the module itself, see the module's dedicated chapter.

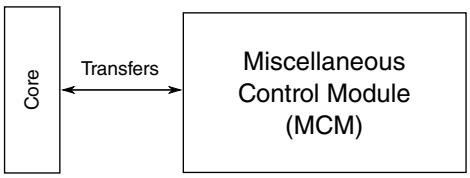


Figure 3-3. MCM configuration

Table 3-3. Reference links to related information

Topic	Related module	Reference
Full description	Miscellaneous Control Module (MCM)	MCM
System memory map		System memory map
Clocking		Clock Distribution
Power management		Power management

3.3.3 Inter-Peripheral Crossbar Switch (XBAR) and AND/OR/INVERT (AOI) Configuration

This section summarizes how the module has been configured in the chip. For a comprehensive description of the module itself, see the module's dedicated chapter.

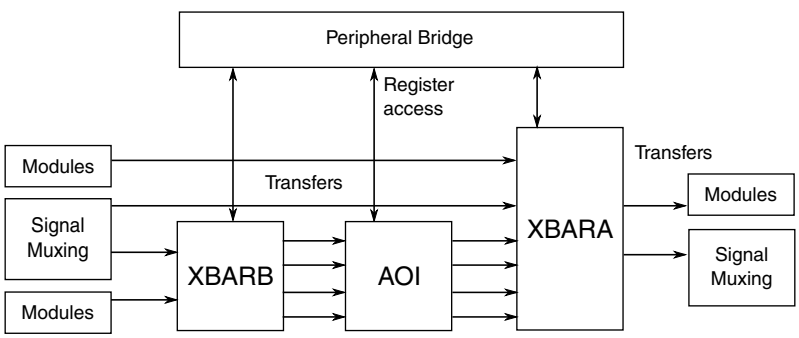


Figure 3-4. XBARA, XBARB, and AOI integration

Table 3-4. Reference links to related information

Topic	Related module	Reference
Full description	XBARA	Inter-Peripheral Crossbar Switch A (XBARA)
Full description	XBARB	Inter-Peripheral Crossbar Switch B (XBARB)
Full description	AOI	Inter-Peripheral Crossbar AND/OR/INVERT (AOI) module
System memory map		System memory map
Clocking		Clock Distribution
Power management		Power management
Signal multiplexing	SIM's GPS registers	Signal multiplexing

3.3.3.1 Number of inputs and outputs

The dedicated XBAR chapters refer to each instance's number of inputs as NUM_IN or N and number of outputs as NUM_OUT or M. The following table identifies the values.

Table 3-5. XBAR inputs and outputs

XBAR instance	Number of inputs (NUM_IN or N)	Number of outputs (NUM_OUT or M)
XBARA	50	59
XBARB	26	16

3.3.3.2 XBARA and XBARB Inputs

The following table shows the signals that can be inputs to the XBARs.

Table 3-6. XBARA and XBARB Inputs

Package Signal	Signal Description	XBARA Input	XBARB Input
VSS	VSS	XBAR_IN0	—
VDD	VDD	XBAR_IN1	—
XB_IN2	Package Pin	XBAR_IN2	XBAR_IN14
XB_IN3	Package Pin	XBAR_IN3	XBAR_IN15
XB_IN4	Package Pin	XBAR_IN4	—
XB_IN5	Package Pin	XBAR_IN5	—
XB_IN6	Package Pin	XBAR_IN6	—
XB_IN7	Package Pin	XBAR_IN7	—
XB_IN8	Package Pin	XBAR_IN8	—
XB_IN9	Package Pin	XBAR_IN9	—
XB_IN10	Package Pin	XBAR_IN10	XBAR_IN20
XB_IN11	Package Pin	XBAR_IN11	XBAR_IN21
CMPA_OUT	Comparator A Output	XBAR_IN12	XBAR_IN0
CMPB_OUT	Comparator B Output	XBAR_IN13	XBAR_IN1
CMPC_OUT	Comparator C Output	XBAR_IN14	XBAR_IN2
CMPD_OUT	Comparator D Output	XBAR_IN15	XBAR_IN3
TB0_OUT	TimerB 0	XBAR_IN16	XBAR_IN4
TB1_OUT	TimerB 1	XBAR_IN17	XBAR_IN5
TB2_OUT	TimerB 2	XBAR_IN18	XBAR_IN6
TB3_OUT	TimerB 3	XBAR_IN19	XBAR_IN7
PWMA0_TRG0	PWMA0 Trigger 0	XBAR_IN20	XBAR_IN8
PWMA0_TRG1	PWMA0 Trigger 1	XBAR_IN21	(PWMA0_TRG0 | PWMA0_TRG1)

Table continues on the next page...

Table 3-6. XBARA and XBARB Inputs (continued)

Package Signal	Signal Description	XBARA Input	XBARB Input
PWMA1_TRG0	PWMA1 Trigger 0	XBAR_IN22	XBAR_IN9 (PWMA1_TRG0 | PWMA1_TRG1)
PWMA1_TRG1	PWMA1 Trigger 1	XBAR_IN23	
PWMA2_TRG0	PWMA2 Trigger 0	XBAR_IN24	XBAR_IN10 (PWMA2_TRG0 | PWMA2_TRG1)
PWMA2_TRG1	PWMA2 Trigger 1	XBAR_IN25	
PWMA3_TRG0	PWMA3 Trigger 0	XBAR_IN26	XBAR_IN11 (PWMA3_TRG0 | PWMA3_TRG1)
PWMA3_TRG1	PWMA3 Trigger 1	XBAR_IN27	
PDB0_OUT_A	PDB0 Trigger Output A	XBAR_IN28	—
PDB0_OUT_B	PDB0 Trigger Output B	XBAR_IN29	XBAR_IN12 (PDB0_OUT_B | PDB0_OUT_D)
PDB0_OUT_C	PDB0 Trigger Output C	XBAR_IN30	—
PDB0_OUT_D	PDB0 Trigger Output D	XBAR_IN31	XBAR_IN12 (PDB0_OUT_B | PDB0_OUT_D)
PDB1_OUT_A	PDB1 Trigger Output A	XBAR_IN32	—
PDB1_OUT_B	PDB1 Trigger Output B	XBAR_IN33	XBAR_IN13 (PDB1_OUT_B | PDB1_OUT_D)
PDB1_OUT_C	PDB1 Trigger Output C	XBAR_IN34	—
PDB1_OUT_D	PDB1 Trigger Output D	XBAR_IN35	XBAR_IN13 (PDB1_OUT_B | PDB1_OUT_D)
TA0_OUT	TimerA 0	XBAR_IN36	XBAR_IN16
TA1_OUT	TimerA 1	XBAR_IN37	XBAR_IN17
TA2_OUT	TimerA 2	XBAR_IN38	XBAR_IN18
TA3_OUT	TimerA 3	XBAR_IN39	XBAR_IN19
AND_OR_INVERT_0	AOI Output 0	XBAR_IN46	—
AND_OR_INVERT_1	AOI Output 1	XBAR_IN47	—
AND_OR_INVERT_2	AOI Output 2	XBAR_IN48	—
AND_OR_INVERT_3	AOI Output 3	XBAR_IN49	—

3.3.3.3 XBAR Interconnections

The following table shows how, within the chip, the AOI module mediates between XBARB outputs and XBARA inputs.

Table 3-7. XBAR Interconnections

XBARB Output	AOI Input/Output	XBARA Input
XBAR_OUT0	AND_OR_INVERT_0	XBAR_IN46
XBAR_OUT1		
XBAR_OUT2		
XBAR_OUT3		
XBAR_OUT4	AND_OR_INVERT_1	XBAR_IN47
XBAR_OUT5		
XBAR_OUT6		
XBAR_OUT7		
XBAR_OUT8	AND_OR_INVERT_2	XBAR_IN48
XBAR_OUT9		
XBAR_OUT10		
XBAR_OUT11		
XBAR_OUT12	AND_OR_INVERT_3	XBAR_IN49
XBAR_OUT13		
XBAR_OUT14		
XBAR_OUT15		

3.3.3.4 XBARA Outputs

Table 3-8. XBARA Outputs

XBARA Output	Signal	Signal Description
XBAR_OUT0	DMA_REQ0	XBAR DMA Request 0
XBAR_OUT1	DMA_REQ1	XBAR DMA Request 1
XBAR_OUT2	DMA_REQ2	XBAR DMA Request 2
XBAR_OUT3	DMA_REQ3	XBAR DMA Request 3
XBAR_OUT4	XB_OUT4	Package Pin
XBAR_OUT5	XB_OUT5	Package Pin
XBAR_OUT6	XB_OUT6	Package Pin
XBAR_OUT7	XB_OUT7	Package Pin
XBAR_OUT8	XB_OUT8	Package Pin
XBAR_OUT9	XB_OUT9	Package Pin
XBAR_OUT10	XB_OUT10	Package Pin
XBAR_OUT11	XB_OUT11	Package Pin

Table continues on the next page...

Table 3-8. XBARA Outputs (continued)

XBARA Output	Signal	Signal Description
XBAR_OUT12	ADCA_TRIG	ADCA (Cyclic ADC) Trigger
XBAR_OUT13	ADCB_TRIG	ADCB (Cyclic ADC) Trigger
XBAR_OUT14	ADCC_TRIG	ADCC (SAR ADC) Trigger
XBAR_OUT15	DAC_12B_SYNC	12-bit DAC Synch
XBAR_OUT16	CMPA	Comparator A Window/Sample
XBAR_OUT17	CMPB	Comparator B Window/Sample
XBAR_OUT18	CMPC	Comparator C Window/Sample
XBAR_OUT19	CMPD	Comparator D Window/Sample
XBAR_OUT20	PWMA0_EXTA	PWMA0 external input
XBAR_OUT21	PWMA1_EXTA	PWMA1 external input
XBAR_OUT22	PWMA2_EXTA	PWMA2 external input
XBAR_OUT23	PWMA3_EXTA	PWMA3 external input
XBAR_OUT24	PWMA0_EXT_SYNC	PWMA0 Ext Synch
XBAR_OUT25	PWMA1_EXT_SYNC	PWMA1 Ext Synch
XBAR_OUT26	PWMA2_EXT_SYNC	PWMA2 Ext Synch
XBAR_OUT27	PWMA3_EXT_SYNC	PWMA3 Ext Synch
XBAR_OUT28	PWMA_EXT_CLK	PWMA External Clock
XBAR_OUT29	PWMA_FAULT0	PWMA Fault0
XBAR_OUT30	PWMA_FAULT1	PWMA Fault1
XBAR_OUT31	PWMA_FAULT2	PWMA Fault2
XBAR_OUT32	PWMA_FAULT3	PWMA Fault3
XBAR_OUT33	PWMA_FORCE	PWMA Force
XBAR_OUT34	TB0_IN	Timer B0
XBAR_OUT35	TB1_IN	Timer B1
XBAR_OUT36	TB2_IN	Timer B2
XBAR_OUT37	TB3_IN	Timer B3
XBAR_OUT38	PDB0_IN_TRIG0	PDB0 Input Trigger0
XBAR_OUT39	PDB0_FAULTA	PDB0 Fault A
XBAR_OUT40	PDB0_FAULTC	PDB0 Fault C
XBAR_OUT41	PDB1_IN_TRIG0	PDB1 Input Trigger0
XBAR_OUT42	PDB1_FAULTA	PDB1 Fault A
XBAR_OUT43	PDB1_FAULTC	PDB1 Fault C
XBAR_OUT44	—	—
XBAR_OUT45	—	—
XBAR_OUT46	—	—
XBAR_OUT47	—	—
XBAR_OUT48	—	—
XBAR_OUT49	TA0_IN	Timer A0
XBAR_OUT50	TA1_IN	TimerA1

Table continues on the next page...

Table 3-8. XBARA Outputs (continued)

XBARA Output	Signal	Signal Description
XBAR_OUT51	TA2_IN	Timer A2
XBAR_OUT52	TA3_IN	Timer A3
XBAR_OUT53	—	—
XBAR_OUT54	—	—
XBAR_OUT55	—	—
XBAR_OUT56	—	—
XBAR_OUT57	—	—
XBAR_OUT58	EWM_IN	External Watchdog Monitor

3.3.3.5 AOI module register write protection

The AOI module's registers can be write protected.

3.3.4 Interrupt Controller (INTC) Configuration

This section summarizes how the module has been configured in the chip. For a comprehensive description of the module itself, see the module's dedicated chapter.

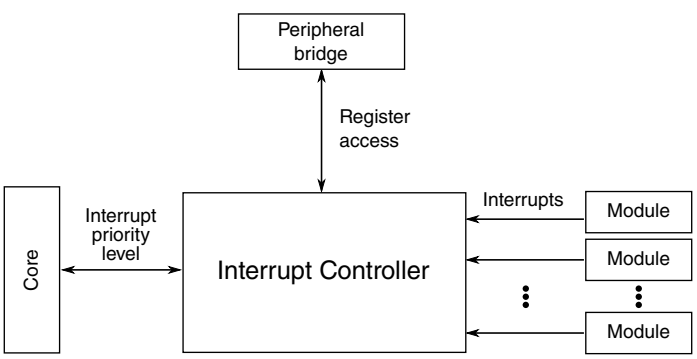


Figure 3-5. INTC configuration

Table 3-9. Reference links to related information

Topic	Related module	Reference
Full description	Interrupt Controller (INTC)	INTC
System memory map		System memory map
Clocking		Clock Distribution
Power management		Power management

3.3.4.1 Reset/Interrupt Vector Table

Table 3-10. Interrupt Vector Table

Module	Vect. #	Priority Level	Vector Base Address	Interrupt	Details	Local Enable	Local Source	Description
Core	110	-1	0xDC	SWILP		N/A	N/A	SWILP
EWM	109	0-2	0xDA	EWM_INT		CTRL[INEN]	N/A	EWM_Indicate
COP	108	0-2	0xD8	COP_INT		INTEN	N/A	COP_Warning
GPIO A	107	0-2	0xD6	GPIOA		IENR	IPEND	GPIO
GPIO B	106	0-2	0xD4	GPIOB		IENR	IPEND	GPIO
GPIO C	105	0-2	0xD2	GPIOC		IENR	IPEND	GPIO
GPIO D	104	0-2	0xD0	GPIOD		IENR	IPEND	GPIO
GPIO E	103	0-2	0xCE	GPIOE		IENR	IPEND	GPIO
GPIO F	102	0-2	0xCC	GPIOF		IENR	IPEND	GPIO
RESERVED	101	0-2	0xCA	—	—	—	—	—
RESERVED	100	0-2	0xC8	—	—	—	—	—
RESERVED	99	0-2	0xC6	—	—	—	—	—
					—	—	—	—
RESERVED	98	0-2	0xC4	—	—	—	—	—
					—	—	—	—
					—	—	—	—
PDB 0	97	0-2	0xC2	PDB0	PDB0_DELAYA	CTRL[DAIE]	CTRL[DAF]	Delay A
					PDB0_DELAYB	CTRL[DBIE]	CTRL[DBF]	Delay B
					PDB0_DELAYC	CTRL[DCIE]	CTRL[DCF]	Delay C
					PDB0_DELAYD	CTRL[DDIE]	CTRL[DDF]	Delay D
					PDB0_OVFLW	CTRL[COIE]	CTRL[COF]	Counter Overflow
PDB 1	96	0-2	0xC0	PDB1	PDB1_DELAYA	CTRL[DAIE]	CTRL[DAF]	Delay A
					PDB1_DELAYB	CTRL[DBIE]	CTRL[DBF]	Delay B
					PDB1_DELAYC	CTRL[DCIE]	CTRL[DCF]	Delay C
					PDB1_DELAYD	CTRL[DDIE]	CTRL[DDF]	Delay D
					PDB1_OVFLW	CTRL[COIE]	CTRL[COF]	Counter Overflow

Table continues on the next page...

Table 3-10. Interrupt Vector Table (continued)

Module	Vect. #	Priority Level	Vector Base Address	Interrupt	Details	Local Enable	Local Source	Description
PIT 0	95	0-2	0xBE	PIT0_ROLLOVR		CTRL[PRIE]	CTRL[PRF]	Roll Over
PIT 1	94	0-2	0xBC	PIT1_ROLLOVR		CTRL[PRIE]	CTRL[PRF]	Roll Over
CMP A	93	0-2	0xBA	CMPA	CMPA_RISE	SCR[IER]	SCR[CFR]	Rising Edge
					CMPA_FALL	SCR[IEF]	SCR[CFF]	Falling Edge
CMP B	92	0-2	0xB8	CMPB	CMPB_RISE	SCR[IER]	SCR[CFR]	Rising Edge
					CMPB_FALL	SCR[IEF]	SCR[CFF]	Falling Edge
CMP C	91	0-2	0xB6	CMPC	CMPC_RISE	SCR[IER]	SCR[CFR]	Rising Edge
					CMPC_FALL	SCR[IEF]	SCR[CFF]	Falling Edge
CMP D	90	0-2	0xB4	CMPD	CMPD_RISE	SCR[IER]	SCR[CFR]	Rising Edge
					CMPD_FALL	SCR[IEF]	SCR[CFF]	Falling Edge
FTFL	89	0-2	0xB2	FTFL_CC		FCNFG[CCIE]	FSTAT[CCIF]	Command Complete
FTFL	88	0-2	0xB0	FTFL_RDCOL		FCNFG[RDCOLIE]	FSTAT[RDCOLERR]	Access Error
eFlex PWM A	87	0-2	0xAE	eFlexPWMA_CMP0		SM0INTEN[CMPIE]	SM0STS[CMPF]	Submodule 0 Compare
eFlex PWM A	86	0-2	0xAC	eFlexPWMA_RELOAD0		SM0INTEN[RIE]	SM0STS[RF	Submodule 0 Reload
eFlex PWM A	85	0-2	0xAA	eFlexPWMA_CMP1		SM1INTEN[CMPIE]	SM1STS[CMPF]	Submodule 1 Compare
eFlex PWM A	84	0-2	0xA8	eFlexPWMA_RELOAD1		SM1INTEN[RIE]	SM1STS[RF	Submodule 1 Reload
eFlex PWM A	83	0-2	0xA6	eFlexPWMA_CMP2		SM2INTEN[CMPIE]	SM2STS[CMPF]	Submodule 2 Compare
eFlex PWM A	82	0-2	0xA4	eFlexPWMA_RELOAD2		SM2INTEN[RIE]	SM2STS[RF	Submodule 2 Reload
eFlex PWM A	81	0-2	0xA2	eFlexPWMA_CMP3		SM3INTEN[CMPIE]	SM3STS[CMPF]	Submodule 3 Compare
eFlex PWM A	80	0-2	0xA0	eFlexPWMA_CAP		SM0INTEN[CFA1IE], SM0INTEN[CFA0IE], SM0INTEN[CFB1IE], SM0INTEN[CFB0IE],	SM0STS[CFA1], SM0STS[CFA0], SM0STS[CFB1], SM0STS[CFB0],	Logic OR of all Submodule 0-3 Input Captures

Table continues on the next page...

Table 3-10. Interrupt Vector Table (continued)

Module	Vect. #	Priority Level	Vector Base Address	Interrupt	Details	Local Enable	Local Source	Description
						SM0INTEN[CFX1IE], SM0INTEN[CFX0IE], SM1INTEN[CFA1IE], SM1INTEN[CFA0IE], SM1INTEN[CFB1IE], SM1INTEN[CFB0IE], SM1INTEN[CFX1IE], SM1INTEN[CFX0IE], SM2INTEN[CFA1IE], SM2INTEN[CFA0IE], SM2INTEN[CFB1IE], SM2INTEN[CFB0IE], SM2INTEN[CFX1IE], SM2INTEN[CFX0IE], SM3INTEN[CFA1IE], SM3INTEN[CFA0IE], SM3INTEN[CFB1IE], SM3INTEN[CFB0IE], SM3INTEN[CFX1IE], SM3INTEN[CFX0IE]	SM0STS[CFX1], SM0STS[CFX0], SM1STS[CFA1], SM1STS[CFB1], SM1STS[CFB0], SM1STS[CFA0], SM1STS[CFX1], SM1STS[CFX0], SM2STS[CFA1], SM2STS[CFA0], SM2STS[CFB1], SM2STS[CFB0], SM2STS[CFX1], SM2STS[CFX0], SM3STS[CFA1], SM3STS[CFA0], SM3STS[CFB1], SM3STS[CFB0], SM3STS[CFX1], SM3STS[CFX0]	
eFlex PWM A	79	0-2	0x9E	eFlexPWMA_RELOAD3		SM3INTEN[RIE]	SM3STS[RF]	Submodule 3 Reload
eFlex PWM A	78	0-2	0x9C	eFlexPWMA_RERR		SM0INTEN[REIE], SM1INTEN[REIE], SM2INTEN[REIE], SM3INTEN[REIE]	SM0STS[REF], SM1STS[REF], SM2STS[REF], SM3STS[REF]	Reload Error
eFlex PWM A	77	0-2	0x9A	eFlexPWMA_FAULT		FCTRL[FIE]	FSTS[FFLAG]	Fault Condition
RESERVED	76	0-2	0x98	—		—	—	—
RESERVED	75	0-2	0x96	—		—	—	—
RESERVED	74	0-2	0x94	—		—	—	—
RESERVED	73	0-2	0x92	—		—	—	—
RESERVED	72	0-2	0x90	—		—	—	—
RESERVED	71	0-2	0x8E	—		—	—	—
RESERVED	70	0-2	0x8C	—		—	—	—
RESERVED	69	0-2	0x8A	—		—	—	—
RESERVED	68	0-2	0x88	—		—	—	—

Table continues on the next page...

Table 3-10. Interrupt Vector Table (continued)

Module	Vect. #	Priority Level	Vector Base Address	Interrupt	Details	Local Enable	Local Source	Description
RESERVED	67	0-2	0x86	—		—	—	—
RESERVED	66	0-2	0x84	—		—	—	—
RESERVED	65	0-2	0x82	—		—	—	—
RESERVED	64	0-2	0x80	—		—	—	—
RESERVED	63	0-2	0x7E	—		—	—	—
IIC 0	62	0-2	0x7C	IIC0		CR1[IICIE]	SR[IICIF]	Complete 1-byte transfer (TCF), Match of received calling address (IAAS), Arbitration Lost (ARBL), SMBus Timeout (SLTF) Interrupts
IIC 1	61	0-2	0x7A	IIC1		CR1[IICIE]	SR[IICIF]	Complete 1-byte transfer (TCF), Match of received calling address (IAAS), Arbitration Lost (ARBL), SMBus Timeout (SLTF) Interrupts
QSPI 0	60	0-2	0x78	QSPI0_RCV		SCTRL[SPRIE]	SCTRL[SPRF, MODF, OVRF]	Receiver Full
	59	0-2	0x76	QSPI0_XMIT		SPSCR[SPTIE]	SPSCR[SPTE]	Transmitter Empty
RESERVED	58	0-2	0x74	—		—	—	—
	57	0-2	0x72	—		—	—	—
RESERVED	56	0-2	0x70	—		—	—	—
	55	0-2	0x6E	—		—	—	—
QSCI 0	54	0-2	0x6C	QSCI0_TDRE		CTRL1[TEIE]	STAT[TDRE]	Transmit Data Register Empty
	53	0-2	0x6A	QSCI0_TIDLE		CTRL1[TIIE]	STAT[TIDLE]	Transmitter Idle
	52	0-2	0x68	QSCI0_RCV		CTRL1[RFIE]	STAT[RDRF, OR, RIEF]	Receive Data Register Full / Overrun / Active Edge
	51	0-2	0x66	QSCI0_RERR		CTRL1[REIE]	STAT[OR, NF, FE, PF, LSE]	Receiver Error

Table continues on the next page...

Table 3-10. Interrupt Vector Table (continued)

Module	Vect. #	Priority Level	Vector Base Address	Interrupt	Details	Local Enable	Local Source	Description
QSCI 1	50	0-2	0x64	QSCI1_TDRE		CTRL1[TEIE]	STAT[TDRE]	Transmit Data Register Empty
	49	0-2	0x62	QSCI1_TIDLE		CTRL1[TIIE]	STAT[TIDLE]	Transmitter Idle
	48	0-2	0x60	QSCI1_RCV		CTRL1[RFIE]	STAT[RDRF, OR, RIEF]	Receive Data Register Full / Overrun / Active Edge
	47	0-2	0x5E	QSCI1_RERR		CTRL1[REIE]	STAT[OR, NF, FE, PF, LSE]	Receiver Error
RESERVED	46	0-2	0x5C	—		—	—	—
	45	0-2	0x5A	—		—	—	—
	44	0-2	0x58	—		—	—	—
	43	0-2	0x56	—		—	—	—
Flex CAN	42	0-2	0x54	WAKEUP		MCR[WAK_MSK]	ESR1[WAK_INT]	Wakeup
Flex CAN	41	0-2	0x52	RX_WARN		CTRL1[RWRN_MSK]	ESR1[RWRN_INT]	Receive Warning
Flex CAN	40	0-2	0x50	TX_WARN		CTRL1[TWRN_MSK]	ESR1[TWRN_INT]	Transmit Warning
Flex CAN	39	0-2	0x4E	ERROR		CTRL1[ERR_MSK]	ESR1[ERR_INT]	Error
Flex CAN	38	0-2	0x4C	BUS_OFF		CTRL1[BOFF_MSK]	ESR1[BOFF_INT]	Bus Off
Flex CAN	37	0-2	0x4A	MB_OR (or all MB bits)		IFLAG1[BUFn]	IFLAG1[BUFn]	Message Buffer
DMA 0	36	0-2	0x48	DMA0		DCR0[EINT]	DSR0[DONE]	DMA 0 Service Req
DMA 1	35	0-2	0x46	DMA1		DCR1[EINT]	DSR1[DONE]	DMA 1 Service Req
DMA 2	34	0-2	0x44	DMA2		DCR2[EINT]	DSR2[DONE]	DMA 2 Service Req
DMA 3	33	0-2	0x42	DMA3		DCR3[EINT]	DSR3[DONE]	DMA 3 Service Req
ADC, SAR	32	0-2	0x40	COCO		ADCSC1n[AIENn]	SC1n[COCOn]	ADC Conversion Complete
ADC, Cyclic	31	0-2	0x3E	ADC_ERR		CTRL1[ZCIE, LLMTIE, HLMTIE]	STAT[ZCI, LLMT, HLMT]	ADC zero crossing, low limit, and high limit
	30	0-2	0x3C	ADC_CC0		CTRL1[EOSIE0]	STAT[EOSI0]	ADC Conversion Complete, any scan type except Converter B in

Table continues on the next page...

Table 3-10. Interrupt Vector Table (continued)

Module	Vect. #	Priority Level	Vector Base Address	Interrupt	Details	Local Enable	Local Source	Description
								nonsimultaneous parallel scan mode
	29	0-2	0x3A	ADC_CC1		CTRL2[EOSIE1]	STAT[EOSi1]	ADC Conversion Complete, Converter B in nonsimultaneous parallel scan mode
TIMER A0	28	0-2	0x38	TMRA_0	TMRA0 TCF	SCTRL[TCFIE]	SCTRL[TCFIE]	Timer Compare
					TMRA0 TOF	SCTRL[TOFIE]	SCTRL[TOF]	Timer Overflow
					TMRA0IEF	SCTRL[IEFIE]	SCTRL[IEF]	Input Edge Flag
					TMRA0 TCF1	CSCTRL[TCF1EN]	CSCTRL[TCF1]	Timer Compare Flag 1
					TMRA0 TCF2	CSCTRL[TCF2EN]	CSCTRL[TCF2]	Timer Compare Flag 2
TIMER A1	27	0-2	0x36	TMRA_1	TMRA1 TCF	SCTRL[TCFIE]	SCTRL[TCFIE]	Timer Compare
					TMRA1 TOF	SCTRL[TOFIE]	SCTRL[TOF]	Timer Overflow
					TMRA1IEF	SCTRL[IEFIE]	SCTRL[IEF]	Input Edge Flag
					TMRA1 TCF1	CSCTRL[TCF1EN]	CSCTRL[TCF1]	Timer Compare Flag 1
					TMRA1 TCF2	CSCTRL[TCF2EN]	CSCTRL[TCF2]	Timer Compare Flag 2
TIMER A2	26	0-2	0x34	TMRA_2	TMRA2 TCF	SCTRL[TCFIE]	SCTRL[TCFIE]	Timer Compare
					TMRA2 TOF	SCTRL[TOFIE]	SCTRL[TOF]	Timer Overflow
					TMRA2IEF	SCTRL[IEFIE]	SCTRL[IEF]	Input Edge Flag
					TMRA2 TCF1	CSCTRL[TCF1EN]	CSCTRL[TCF1]	Timer Compare Flag 1
					TMRA2 TCF2	CSCTRL[TCF2EN]	CSCTRL[TCF2]	Timer Compare Flag 2
TIMER A3	25	0-2	0x32	TMRA_3	TMRA3 TCF	SCTRL[TCFIE]	SCTRL[TCFIE]	Timer Compare
					TMRA3 TOF	SCTRL[TOFIE]	SCTRL[TOF]	Timer Overflow
					TMRA3IEF	SCTRL[IEFIE]	SCTRL[IEF]	Input Edge Flag

Table continues on the next page...

Table 3-10. Interrupt Vector Table (continued)

Module	Vect. #	Priority Level	Vector Base Address	Interrupt	Details	Local Enable	Local Source	Description
					TMRA3 TCF1	CSCTRL[TCF1EN]	CSCTRL[TCF1]	Timer Compare Flag 1
					TMRA3 TCF2	CSCTRL[TCF2EN]	CSCTRL[TCF2]	Timer Compare Flag 2
TIMER B0	24	0-2	0x30	TMRB_0	TMRB0 TCF	SCTRL[TCFIE]	SCTRL[TCFIE]	Timer Compare
					TMRB0 TOF	SCTRL[TOFIE]	SCTRL[TOF]	Timer Overflow
					TMRB0IEF	SCTRL[IEFIE]	SCTRL[IEF]	Input Edge Flag
					TMRB0 TCF1	CSCTRL[TCF1EN]	CSCTRL[TCF1]	Timer Compare Flag 1
					TMRB0 TCF2	CSCTRL[TCF2EN]	CSCTRL[TCF2]	Timer Compare Flag 2
TIMER B1	23	0-2	0x2E	TMRB_1	TMRB1 TCF	SCTRL[TCFIE]	SCTRL[TCFIE]	Timer Compare
					TMRB1 TOF	SCTRL[TOFIE]	SCTRL[TOF]	Timer Overflow
					TMRB1IEF	SCTRL[IEFIE]	SCTRL[IEF]	Input Edge Flag
					TMRB1 TCF1	CSCTRL[TCF1EN]	CSCTRL[TCF1]	Timer Compare Flag 1
					TMRB1 TCF2	CSCTRL[TCF2EN]	CSCTRL[TCF2]	Timer Compare Flag 2
TIMER B2	22	0-2	0x2C	TMRB_2	TMRB2 TCF	SCTRL[TCFIE]	SCTRL[TCFIE]	Timer Compare
					TMRB2 TOF	SCTRL[TOFIE]	SCTRL[TOF]	Timer Overflow
					TMRB2IEF	SCTRL[IEFIE]	SCTRL[IEF]	Input Edge Flag
					TMRB2 TCF1	CSCTRL[TCF1EN]	CSCTRL[TCF1]	Timer Compare Flag 1
					TMRB2 TCF2	CSCTRL[TCF2EN]	CSCTRL[TCF2]	Timer Compare Flag 2
TIMER B3	21	0-2	0x2A	TMRB_3	TMRB3 TCF	SCTRL[TCFIE]	SCTRL[TCFIE]	Timer Compare
					TMRB3 TOF	SCTRL[TOFIE]	SCTRL[TOF]	Timer Overflow
					TMRB3IEF	SCTRL[IEFIE]	SCTRL[IEF]	Input Edge Flag
					TMRB3 TCF1	CSCTRL[TCF1EN]	CSCTRL[TCF1]	Timer Compare Flag 1
					TMRB3 TCF2	CSCTRL[TCF2EN]	CSCTRL[TCF2]	Timer Compare Flag 2

Table continues on the next page...

Table 3-10. Interrupt Vector Table (continued)

Module	Vect. #	Priority Level	Vector Base Address	Interrupt	Details	Local Enable	Local Source	Description
OCCS	20	1-3	0x28	OCCS	OCCSLOLI1	CTRL[PLLIE1]	STAT[LCK1]	PLL Loss of Lock 1
					OCCSLOLI0	CTRL[PLLIE0]	STAT[LCK0]	PLL Loss of Lock 0
					OCCSLOCI	CTRL[LOCIE]	STAT[LOCI]	PLL Loss of Reference Clock
Pwr Supv	19	1-3	0x26	LVI1		CTRL[IOLVIE, CLVIE]	STAT[IOLVS,CLVS]	Low Voltage Interrupt
XBARA	18	1-3	0x24	XBARA		CTRL0[IEN0]| CTRL0[IEN1]| CTRL1[IEN0]| CTRL1[IEN1]	CTRL0[STS0]| CTRL0[STS1]| CTRL1[STS0]| CTRL1[STS1]	Crossbar Interrupt
Core	17	0	0x22	SWI0		N/A	N/A	SW Interrupt 0
Core	16	1	0x20	SWI1		N/A	N/A	SW Interrupt 1
Core	15	2	0x1E	SWI2		N/A	N/A	SW Interrupt 2
Rsrvd	14	1-3	0x1C					
Rsrvd	13	1-3	0x1A					
Rsrvd	12	1-3	0x18					
Core	11	1-3	0x16	BUS_ERR		MCM_CFIER[ECFEI]	Core	Bus Error Interrupt
Core	10	1-3	0x14	RX_REG		IPR0[RX_REG]	Core	EOnCE Receive Register Full
Core	9	1-3	0x12	TX_REG		IPR0[TX_REG]	Core	EOnCE Transmit Register Empty
Core	8	1-3	0x10	TRBUF		IPR0[TRBUF]	Core	EOnCE Trace Buffer Interrupt
Core	7	1-3	0x0E	BKPT		IPR0[BKPT_U]	Core	EOnCE Breakpoint Unit
Core	6	1-3	0x0C	STPCNT		IPR0[STPCNT]	Core	EOnCE Step Counter Interrupt
Core	5	3	0x0A	MISALIGNED		N/A	Core	Misaligned Data Access
Core	4	3	0x08	OVERFLOW		N/A	Core	Hardware Stack Overflow
Core	3	3	0x06	SWI3		N/A	Core	SW Interrupt 3
Core	2	3	0x04	ILLEGAL-OP		N/A	Core	Illegal Instruction
Core	1		0x02	COP_RESET				Reserved for COP Reset Overlay
Core	0		0x00	HW_RESET				Reserved for Reset Overlay

3.3.5 DMA Controller Configuration

This section summarizes how the module has been configured in the chip. For a comprehensive description of the module itself, see the module's dedicated chapter.

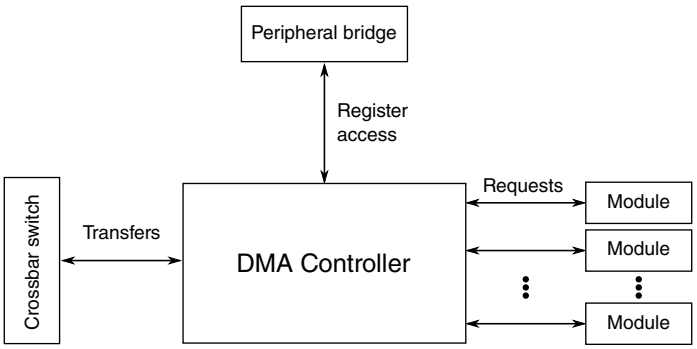


Figure 3-6. DMA Controller configuration

Table 3-11. Reference links to related information

Topic	Related module	Reference
Full description	DMA controller	DMA controller
System memory map		System memory map
Clocking		Clock Distribution
Power management		Power management
Requests		DMA channel assignments

3.3.5.1 DMA channel assignments

The following table identifies the DMA channel assignments.

Table 3-12. DMA Channel Assignment

Chan. Input	Channel 0		Channel 1		Channel 2		Channel 3	
	Signal	Description	Signal	Description	Signal	Description	Signal	Description
0	SCI0_TE	SCI0 Transmitter Empty	SCI1_TE	SCI1 Transmitter Empty	—	—	SCI0_TE	SCI0 Transmitter Empty
1	SCI1_RF	SCI1 Receiver Full	SCI0_RF	SCI0 Receiver Full	SCI0_RF	SCI0 Receiver Full	—	—
2	SPI0_RF	SPI0 Receiver Full	SPI0_TE	SPI0 Transmitter Empty	SPI0_RF	SPI0 Receiver Full	SPI0_TE	SPI0 Transmitter Empty
3	—	—	—	—	—	—	—	—

Table continues on the next page...

Table 3-12. DMA Channel Assignment (continued)

Chan. Input	Channel 0		Channel 1		Channel 2		Channel 3	
	Signal	Description	Signal	Description	Signal	Description	Signal	Description
4	IIC0_ipd_Req	IIC0 DMA Req	IIC1_ipd_Req	IIC1 DMA Req	IIC0_ipd_Req	IIC0 DMA Req	IIC1_ipd_Req	IIC1 DMA Req
5	TMRA0_CP	TMRA0 Capture	TMRA1_CP	TMRA1 Capture	TMRA2_CP	TMRA2 Capture	TMRA3_CP	TMRA3 Capture
6	TMRA0_CMPLD1 | TMRA1_CMPLD2	TMRA0 Compare1 or TMRA1 Compare2	TMRA0_CMPLD2 | TMRA1_CMPLD1	TMRA0 Compare2 or TMRA1 Compare1	TMRA2_CMPLD1 | TMRA3_CMPLD2	TMRA2 Compare1 or TMRA3 Compare2	TMRA2_CMPLD2 | TMRA3_CMPLD1	TMRA2 Compare2 or TMRA3 Compare1
7	TMRB0_CP	TMRB0 Capture	TMRB1_CP	TMRB1 Capture	TMRB2_CP	TMRB2 Capture	TMRB3_CP	TMRB3 Capture
8	TMRB0_CMPLD1 | TMRB1_CMPLD2	TMRB0 Compare1 or TMRB1 Compare2	TMRB0_CMPLD2 | TMRB1_CMPLD1	TMRB0 Compare2 or TMRB1 Compare1	TMRB2_CMPLD1 | TMRB3_CMPLD2	TMRB2 Compare1 or TMRB3 Compare2	TMRB2_CMPLD2 | TMRB3_CMPLD1	TMRB2 Compare2 or TMRB3 Compare1
9	PWMA3_CP	Submodule 3 Capture DMA Req	PWMA2_CP	Submodule 2 Capture DMA Req	PWMA1_CP	Submodule 1 Capture DMA Req	PWMA0_CP	Submodule 0 Capture DMA Req
10	—	—	—	—	—	—	—	—
11	—	—	DAC_FIFO (8 deep)	12bit DAC FIFO Water Mark	PWMA0_WR | PWMA1_WR | PWMA2_WR | PWMA3_WR	SubModule0 Value write DMA Req or SubModule1 Value write DMA Req or SubModule2 Value write DMA Req or SubModule3 Value write DMA Req	DAC_FIFO(8 deep)	12-bit DAC FIFO watermark
12	ADCA_ES (Cyclic)	ADCA End of Scan	ADCA_ES (Cyclic)	ADCA End of Scan	ADCA_ES (Cyclic)	ADCA End of Scan	ADCA_ES (Cyclic)	ADCA End of Scan
13	ADCB_ES (Cyclic)	ADCB End of Scan	ADCB_ES (Cyclic)	ADCB End of Scan	ADCB_ES (Cyclic)	ADCB End of Scan	ADCB_ES (Cyclic)	ADCB End of Scan
14	ADCC_EC (SAR)	ADCC End of Conv.	ADCC_EC (SAR)	ADCC End of Conv.	ADCC_EC (SAR)	ADCC End of Conv.	ADCC_EC (SAR)	ADCC End of Conv.
15	XBAR_DSC0	XBAR DMA Req 0	XBAR_DSC1	XBAR DMA Req 1	XBAR_DSC2	XBAR DMA Req 2	XBAR_DSC3	XBAR DMA Req 3

3.3.6 Power Management Controller (PMC) Configuration

This section summarizes how the module has been configured in the chip. For a comprehensive description of the module itself, see the module's dedicated chapter.

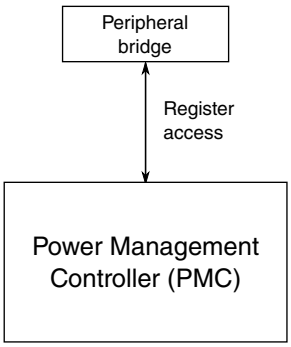


Figure 3-7. PMC configuration

Table 3-13. Reference links to related information

Topic	Related module	Reference
Full description	PMC	PMC
System memory map		System memory map
Clocking		Clock Distribution
Power management		Power management

3.4 Clock Modules

3.4.1 On-Chip Clock Synthesis (OCCS) Configuration

This section summarizes how the module has been configured in the chip. For a comprehensive description of the module itself, see the module's dedicated chapter.

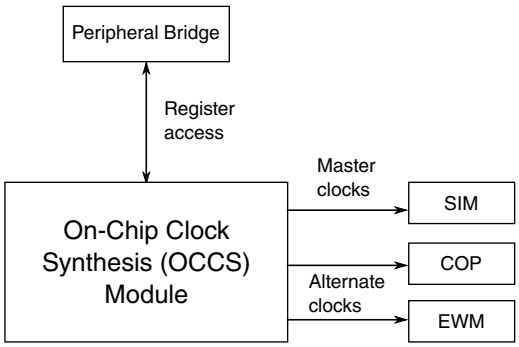


Figure 3-8. OCCS configuration

Table 3-14. Reference links to related information

Topic	Related module	Reference
Full description	On-Chip Clock Synthesis (OCCS)	OCCS
System memory map		System memory map
Clocking		Clock Distribution
Power management		Power management

3.5 Memories and Memory Interfaces

3.5.1 Flash Memory Controller (FMC) Configuration

This section summarizes how the module has been configured in the chip. For a comprehensive description of the module itself, see the module's dedicated chapter.

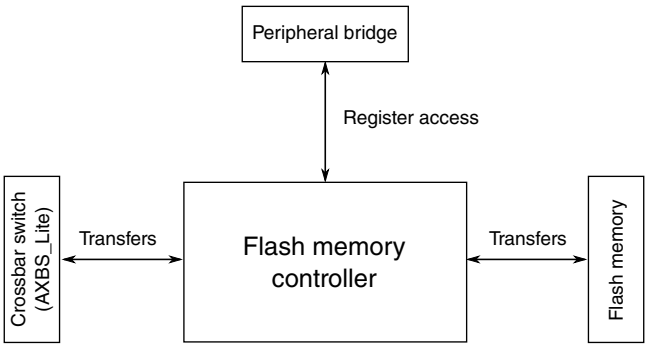


Figure 3-9. Flash memory controller configuration

Table 3-15. Reference links to related information

Topic	Related module	Reference
Full description	Flash memory controller	Flash memory controller
System memory map		System memory map
Clocking		Clock Distribution
Transfers	Flash memory	Flash memory

3.5.2 Flash Memory Configuration

This section summarizes how the module has been configured in the chip. For a comprehensive description of the module itself, see the module's dedicated chapter.

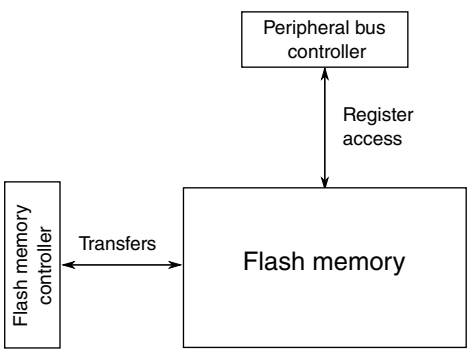


Figure 3-10. Flash memory configuration

Table 3-16. Reference links to related information

Topic	Related module	Reference
Full description	Flash memory	Flash memory module (FTFL)
System memory map		System memory map
Clocking		Clock Distribution
Power management		Power management

3.5.2.1 Flash memory types and terminology

This device contains the following types of flash memory:

- Program flash memory — non-volatile flash memory that can execute program code or store data
- FlexMemory — encompasses the following memory types:
  - FlexNVM — Non-volatile flash memory that can execute program code, store data, or back up EEPROM data
  - FlexRAM — RAM memory that can be used as traditional RAM or as high-endurance EEPROM storage in conjunction with FlexNVM, and also accelerates flash memory programming

The following table explains how these types of flash memory align with terminology used in other descriptions of the flash memory.

Table 3-17. Flash memory terminology

Flash memory type	Memory Map	Flash Memory Module (FTFL)
Program flash memory	Primary program/data flash memory	Program flash memory
FlexNVM	Secondary program/data flash memory	Data flash memory (as well as FlexNVM)

3.5.2.2 FTFL_FOPT Register

The flash memory's FTFL_FOPT register allows the user to customize the operation of the MCU at boot time. See FOPT boot options for details of its definition.

3.6 Security and Integrity

3.6.1 Computer Operating Properly (COP) Module Configuration

This section summarizes how the module has been configured in the chip. For a comprehensive description of the module itself, see the module's dedicated chapter.

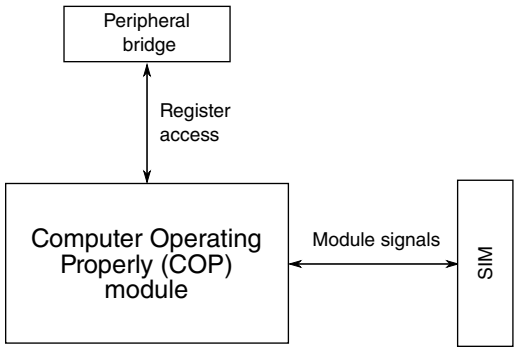


Figure 3-11. COP configuration

Table 3-18. Reference links to related information

Topic	Related module	Reference
Full description	Computer Operating Properly (COP) module	COP
System memory map		System memory map
Clocking		Clock Distribution
Power management		Power management
System Integration Module (SIM)	SIM's RSTAT register	SIM

3.6.1.1 COP low power clocks

The COP_CTRL[CLKSEL] bitfield selects among the options for the clock source of the COP module's counter. For each value of the bitfield, the following table correlates between the module-level and chip-level names of the clock options.

Table 3-19. COP counter clock selection

COP_CTRL[CLKSEL] value	COP clock name	Chip clock
00b	ROSC	ROSC_8M (8 MHz ROSC)
01b	COSC	XTAL_OSC (XOSC clock)
10b	Bus clock	Bus clock[1]
11b	Low speed oscillator	ROSC_32K (32 kHz ROSC)

1. Do not select the bus clock to clock the counter if the application requires the COP to wake the device from stop mode.

3.6.2 External Watchdog Monitor (EWM) Configuration

This section summarizes how the module has been configured in the chip. For a comprehensive description of the module itself, see the module's dedicated chapter.

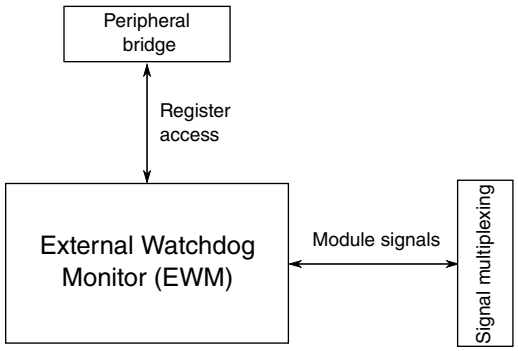


Figure 3-12. External Watchdog Monitor configuration

Table 3-20. Reference links to related information

Topic	Related module	Reference
Full description	External Watchdog Monitor (EWM)	EWM
System memory map		System memory map
Clocking		Clock Distribution
Power management		Power management
Signal multiplexing	SIM's GPS registers	Signal multiplexing

3.6.2.1 EWM low power clocks

The EWM_CLKCTRL[CLKSEL] bitfield selects among the options for the EWM's low power clock source. For each value of the bitfield, the following table correlates between the module-level and chip-level names of the clock options.

Table 3-21. EWM counter clock selection

EWM_CLKCTRL[CLKSEL] value	EWM clock name	Chip clock
00b	lpo_clk[0]	ROSC_8M (8 MHz ROSC)
01b	lpo_clk[1]	XTAL_OSC (XOSC clock)
10b	lpo_clk[2]	Bus clock
11b	lpo_clk[3]	ROSC_32K (32 kHz ROSC)

3.6.2.2 $\overline{\text{EWM_OUT}}$ pin state in Low Power Modes

During Wait and Stop modes, the $\overline{\text{EWM_OUT}}$ pin enters a high-impedance state. A user has the option to control the logic state of the pin using an external pull device or by configuring the internal pull device. When the CPU enters a Run mode from Wait or Stop recovery, the pin resumes the state it had prior to the entry to Wait or Stop mode.

3.6.2.3 EWM_IN signal

The EWM_IN signal is available via the XBARA module's XBAR_OUT40 output.

3.6.3 Cyclic Redundancy Check (CRC) Configuration

This section summarizes how the module has been configured in the chip. For a comprehensive description of the module itself, see the module's dedicated chapter.

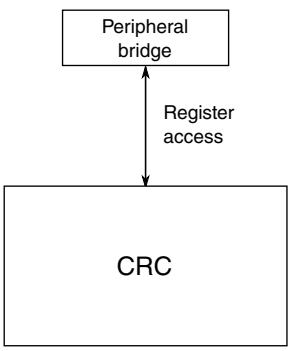


Figure 3-13. CRC configuration

Table 3-22. Reference links to related information

Topic	Related module	Reference
Full description	CRC	CRC
System memory map		System memory map
Power management		Power management

3.7 Analog

3.7.1 SAR Analog-to-Digital Converter (ADC) Configuration

This section summarizes how the module has been configured in the chip. For a comprehensive description of the module itself, see the module's dedicated chapter.

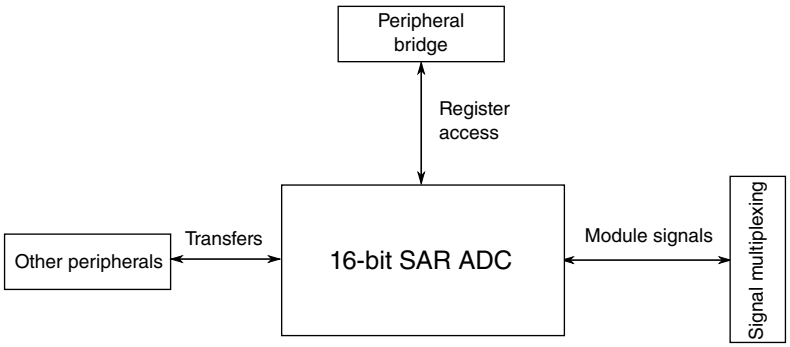


Figure 3-14. SAR ADC configuration

Table 3-23. Reference links to related information

Topic	Related module	Reference
Full description	SAR ADC	16-bit SAR ADC
System memory map		System memory map
Clocking		Clock Distribution
Power management		Power management
Signal multiplexing	SIM's GPS registers	Signal multiplexing

3.7.1.1 SAR ADC Instantiation

The SAR ADC module's mnemonic is ADC16. Its signals are labeled with a C, as in ANC, ADCC, VREFLC, and VREFHC.

3.7.1.2 SAR ADC Channel Assignments

Table 3-24. SAR ADC Channel Assignments

Channel Number	Source
AD0	Reserved
AD1	Reserved
AD2	Reserved
AD3	Reserved
AD4	Reserved
AD5	Reserved
AD6	Reserved
AD7	Reserved
AD8	GPIOA[4]
AD9	GPIOA[5]
AD10	GPIOA[6]

Table continues on the next page...

Table 3-24. SAR ADC Channel Assignments (continued)

Channel Number	Source
AD11	GPIOA[7]
AD12	GPIOB[4]
AD13	GPIOB[5]
AD14	GPIOB[6]
AD15	GPIOB[7]
AD16	GPIOA[8]
AD17	GPIOA[9]
AD18	GPIOA[10]
AD19	GPIOA[11]
AD20	GPIOB[8]
AD21	GPIOB[9]
AD22	GPIOB[10]
AD23	GPIOB[11]
AD24	Reserved
AD25	Reserved
AD26	Temperature Sensor
AD27	Buffered bandgap reference output
AD28	Reserved
AD29	VREFSH
AD30	VREFSL

3.7.1.3 SAR ADC clock in stop modes

For the SAR ADC to operate in any of this device's stop modes, it must use its own internal ASYNC clock, ADACK.

3.7.1.4 SAR ADC alternate clock

For this device, the SAR ADC's alternate clock is the 8 MHz ROSC divided by two.

3.7.1.5 SAR ADC Reference Options

Select the reference using the SAR ADC's SC2[REFSEL] field.

Table 3-25. SAR ADC Reference Options

SC2[REFSEL] value	Reference	Description	Note
00b	VREFH and VREFL	Primary reference option	
01b	VREFHC and VREFLC	External V_{ALT} reference option	Available only on 100-pin and 80-pin packages
10b	VREFBG	Internal V_{ALT} reference option	Buffered output of PMC's 1.2 V bandgap reference
11b	Reserved		

3.7.1.6 SAR ADC Hardware Conversion Trigger Source

The source of the SAR ADC's selectable asynchronous hardware conversion trigger, ADHWT, is XBARA's XBAR_OUT14. See XBARA Outputs.

3.7.2 Cyclic Analog-to-Digital Converter (ADC) Configuration

This section summarizes how the module has been configured in the chip. For a comprehensive description of the module itself, see the module's dedicated chapter.

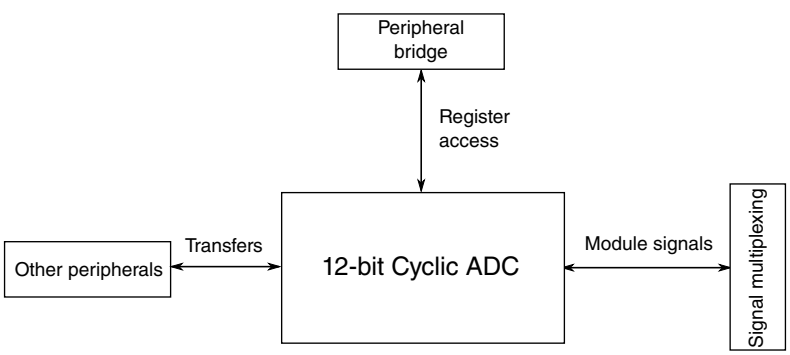


Figure 3-15. Cyclic ADC configuration

Table 3-26. Reference links to related information

Topic	Related module	Reference
Full description	12-bit Cyclic ADC	Cyclic ADC
System memory map		System memory map
Clocking		Clock Distribution
Power management		Power management
Signal multiplexing	SIM's GPS registers	Signal multiplexing

3.7.2.1 Cyclic ADC Instantiation

The cyclic ADC module's mnemonic is ADC12. It is a dual ADC. The signals of its first ADC are labeled A, as in ANA, ADCA, VREFLA, and VREFHA. The signals of its second ADC are labeled B, as in ANB, ADCB, VREFLB, and VREFHB.

3.7.2.2 Cyclic ADC SYNC Signal Connections

The XBARA module's outputs XBAR_OUT12 and XBAR_OUT13 are connected to ADCA's SYNC input and ADCB's SYNC input, respectively. Through these XBARA connections, each ADC can be synchronized to another module, such as a PWM.

3.7.2.3 Cyclic ADC and PWM Connections

Within the chip, the cyclic ADC has internal connections for PWM control.

Table 3-27. Cyclic ADC and PWM Connections

Cyclic ADC Outputs	PWM Inputs
an0_pwm	PWMA0_EXTB
an1_pwm	PWMA1_EXTB
an2_pwm	PWMA2_EXTB
an3_pwm	PWMA3_EXTB

3.7.3 Comparator (CMP) Configuration

This section summarizes how the module has been configured in the chip. For a comprehensive description of the module itself, see the module’s dedicated chapter.

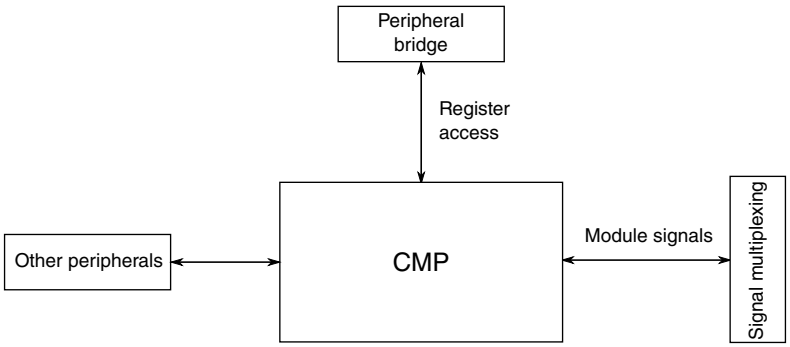


Figure 3-16. CMP Configuration

Table 3-28. Reference links to related information

Topic	Related module	Reference
Full description	Comparator (CMP)	Comparator
System memory map		System memory map
Clocking		Clock Distribution
Power management		Power management
Signal multiplexing	SIM's GPS registers	Signal multiplexing

3.7.3.1 Comparator Channel Assignments

Table 3-29. Comparator Channel Assignments

Module	Comparator/Mux Channel	Source
CMPA	Channel 7	6-bit DAC (CMPA)
	Channel 6	N/C
	Channel 5	GPIOC6
	Channel 4	12-bit DAC
	Channel 3	GPIOA0
	Channel 2	GPIOA3
	Channel 1	GPIOA2
	Channel 0	GPIOA1
CMPB	Channel 7	6-bit DAC (CMPB)
	Channel 6	N/C
	Channel 5	GPIOC6
	Channel 4	12-bit DAC
	Channel 3	GPIOB0
	Channel 2	GPIOB7
	Channel 1	GPIOB6
	Channel 0	GPIOB1
CMPC	Channel 7	6-bit DAC (CMPC)

Table continues on the next page...

Table 3-29. Comparator Channel Assignments (continued)

Module	Comparator/Mux Channel	Source
	Channel 6	N/C
	Channel 5	GPIOC6
	Channel 4	12-bit DAC
	Channel 3	GPIOB2
	Channel 2	GPIOB5
	Channel 1	GPIOB4
	Channel 0	GPIOB3
CMPD	Channel 7	6-bit DAC (CMPD)
	Channel 6	N/C
	Channel 5	GPIOC6
	Channel 4	12-bit DAC
	Channel 3	N/C
	Channel 2	N/C
	Channel 1	N/C
	Channel0	GPIOA4

3.7.3.2 Comparator Voltage References

The 6-bit DAC sub-block supports the selection of two references. For this device, both the Vin1 input and the Vin2 input are connected to a separate V_{DDA} at the package level.

3.7.4 12-bit Digital-to-Analog Converter (DAC) Configuration

This section summarizes how the module has been configured in the chip. For a comprehensive description of the module itself, see the module's dedicated chapter.

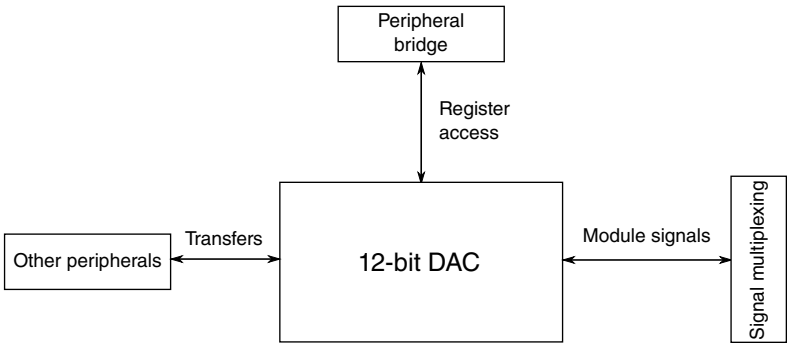


Figure 3-17. 12-bit DAC Configuration

Table 3-30. Reference links to related information

Topic	Related module	Reference
Full description	12-bit DAC	12-bit DAC
System memory map		System memory map
Clocking		Clock Distribution
Power management		Power management
Signal multiplexing	SIM's GPS registers	Signal multiplexing

3.8 Timers and PWM

3.8.1 PWM Configuration

This section summarizes how the module has been configured in the chip. For a comprehensive description of the module itself, see the module's dedicated chapter.

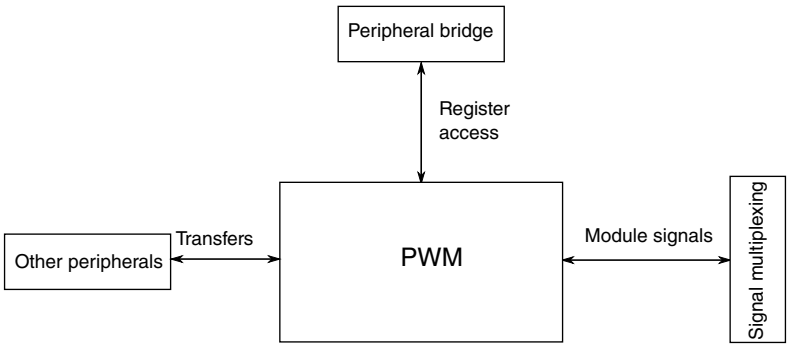


Figure 3-18. PWM Configuration

Table 3-31. Reference links to related information

Topic	Related module	Reference
Full description	PWM	PWM
System memory map		System memory map
Clocking		Clock Distribution
Power management		Power management
Signal multiplexing	SIM's GPS registers	Signal multiplexing

3.8.1.1 PWM auxiliary signals and analog inputs

Cyclic ADC (ADC12) conversion high/low limits can be fed to the PWM auxiliary signals when they are used as inputs. In this configuration:

- When an ADC conversion is higher than a pre-programmed high limit, the signal going to the PWM auxiliary input is low.
- When an ADC conversion is lower than a pre-programmed low limit, the signal going to the PWM auxiliary input is high.

3.8.2 PDB Configuration

This section summarizes how the module has been configured in the chip. For a comprehensive description of the module itself, see the module's dedicated chapter.

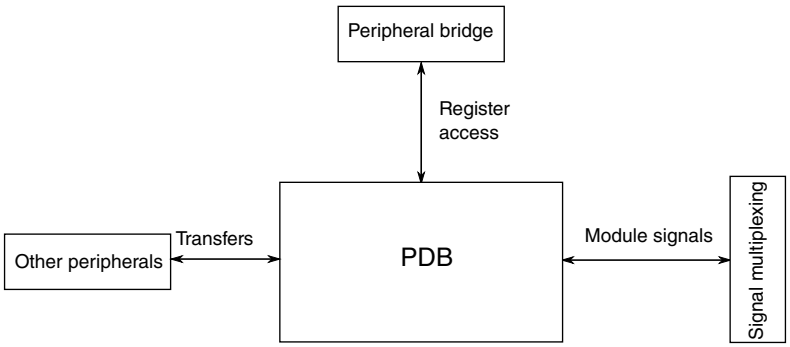


Figure 3-19. PDB Configuration

Table 3-32. Reference links to related information

Topic	Related module	Reference
Full description	PDB	PDB
System memory map		System memory map
Clocking		Clock Distribution
Power management		Power management
Signal multiplexing	SIM's GPS registers	Signal multiplexing

3.8.2.1 PDB Trigger Connections

Table 3-33. PDB Trigger Connections

Signal	Direction	Description	Connection
PDB0_IN_TRIG[6:0]	Input	Trigger input	Trigger[0] from XBARA (remainder are reserved)
PDB1_IN_TRIG[6:0]	Input	Trigger input	Trigger[0] from XBARA (remainder are reserved)
PDB0_pretrig_a_out PDB0_pretrig_b_out PDB0_pretrig_c_out PDB0_pretrig_d_out	Output	Pretrigger output	Reserved
PDB1_pretrig_a_out PDB1_pretrig_b_out PDB1_pretrig_c_out PDB1_pretrig_d_out	Output	Pretrigger output	Reserved
PDB0_OUT_A PDB0_OUT_B PDB0_OUT_C PDB0_OUT_D	Output	Trigger output	To XBARA (all) To XBARB (PDB0_OUT_B | PDB0_OUT_D)
PDB1_OUT_A	Output	Trigger output	To XBARA (all)

Table 3-33. PDB Trigger Connections

Signal	Direction	Description	Connection
PDB1_OUT_B PDB1_OUT_C PDB1_OUT_D			To XBARB (PDB1_OUT_B | PDB1_OUT_D)

3.8.3 PIT Configuration

This section summarizes how the module has been configured in the chip. For a comprehensive description of the module itself, see the module's dedicated chapter.

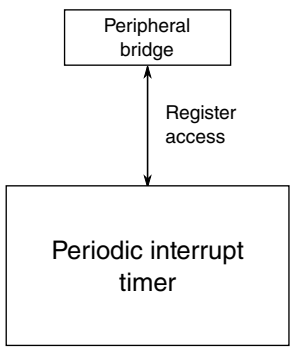


Figure 3-20. PIT configuration

Table 3-34. Reference links to related information

Topic	Related module	Reference
Full description	PIT	PIT
System memory map		System memory map
Clocking		Clock Distribution
Power management		Power management
Signal multiplexing	SIM's GPS registers	Signal multiplexing

3.8.3.1 PIT low power clocks

Each PIT module's CTRL[CLKSEL] bitfield selects among the options for that PIT's counter clock source. For each value of the bitfield, the following table correlates between the module-level and chip-level names of the clock options.

Table 3-35. PIT counter clock selection

PITx_CTRL[CLKSEL] value	PIT clock name	Chip clock
00b	IPBus clock	BUS_CLK
01b	Alternate clock 1	XTAL_OSC (XOSC clock)
10b	Alternate clock 2	ROSC_8M (8 MHz ROSC)
11b	Alternate clock 3	ROSC_32K (32 kHz ROSC)

3.8.3.2 PIT master/slave selection

Either PIT0 or PIT1 can be the master of the other PIT module instance. Use the SIM's MISC0[PIT_MSTR] bit to select between the two master/slave configurations. The master PIT module instance generates the count_enable signal to synchronize both instances of the PIT.

3.8.4 TMR Configuration

This section summarizes how the module has been configured in the chip. For a comprehensive description of the module itself, see the module's dedicated chapter.

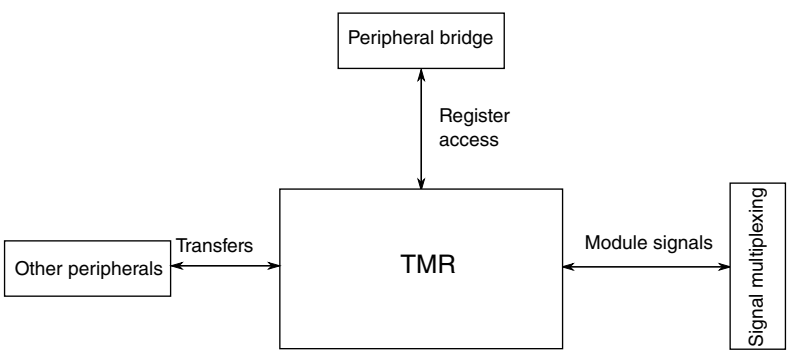


Figure 3-21. TMR Configuration

Table 3-36. Reference links to related information

Topic	Related module	Reference
Full description	TMR	TMR

Table continues on the next page...

Table 3-36. Reference links to related information (continued)

Topic	Related module	Reference
System memory map		System memory map
Clocking		Clock Distribution
Power management		Power management
Signal multiplexing	SIM's GPS registers	Signal multiplexing

3.9 Communication interfaces

3.9.1 CAN Configuration

This section summarizes how the module has been configured in the chip. For a comprehensive description of the module itself, see the module's dedicated chapter.

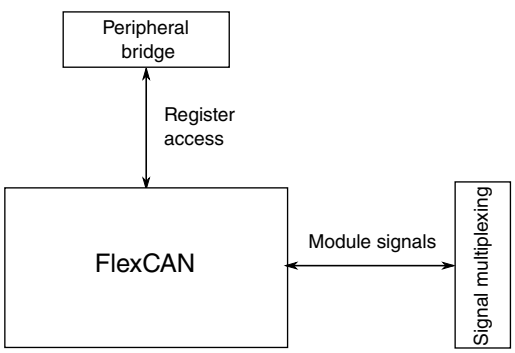


Figure 3-22. CAN configuration

Table 3-37. Reference links to related information

Topic	Related module	Reference
Full description	CAN	CAN
System memory map		System memory map
Clocking		Clock Distribution
Power management		Power management
Signal multiplexing	SIM's GPS registers	Signal multiplexing

3.9.1.1 FlexCAN3 glitch filter

This chip supports wakeup from the FlexCAN3 module's Stop and Doze mode through a CAN wakeup interrupt. Any recessive to dominant transition on the CAN bus (CAN_RX) can wake the chip from Stop or Doze mode. An optional glitch filter is connected on CAN_RX to the interrupt generation logic path.

The glitch filter provides the following functionality:

- Filtering out of unwanted noise on the CAN bus
- Selection of the wakeup source, either from the filtered or unfiltered CAN bus
- Routing of the wakeup source to either the synchronous (Doze) or asynchronous (Stop) wakeup path within the FlexCAN module

The reference clock for the glitch filter is a 4 MHz clock derived from the on-chip 8 MHz ROSC. The 8 MHz ROSC must remain on if the user wants a low power wakeup through the glitch filter. The glitch filter counts 11 cycles of the 4 MHz clock before recognizing it as a valid recessive to dominant transition.

3.9.1.2 FlexCAN3 Supervisor Mode

The module's MCR[SUPV] field configures the FlexCAN to be in either Supervisor or User Mode. On this device:

- MCR[SUPV] is always 1: the FlexCAN is in Supervisor Mode.
- Writes to MCR[SUPV] have no effect.

The device's memory resource protection (MRP) functionality also uses the terms *supervisor* and *user* to refer to an entirely different partitioning of memory regions. In MRP terminology, the FlexCAN's memory space can be defined as a supervisor region to protect it from unauthorized user code.

3.9.2 Serial Peripheral Interface (SPI) Configuration

This section summarizes how the module has been configured in the chip. For a comprehensive description of the module itself, see the module’s dedicated chapter.

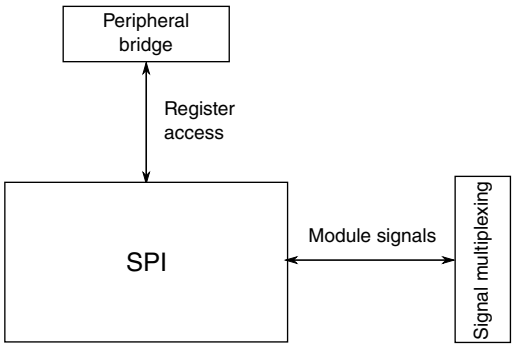


Figure 3-23. SPI configuration

Table 3-38. Reference links to related information

Topic	Related module	Reference
Full description	SPI	SPI
System memory map		System memory map
Clocking		Clock Distribution
Power management		Power management
Signal multiplexing	SIM's GPS registers	Signal multiplexing

3.9.3 Inter-Integrated Circuit (I2C) Configuration

This section summarizes how the module has been configured in the chip. For a comprehensive description of the module itself, see the module's dedicated chapter.

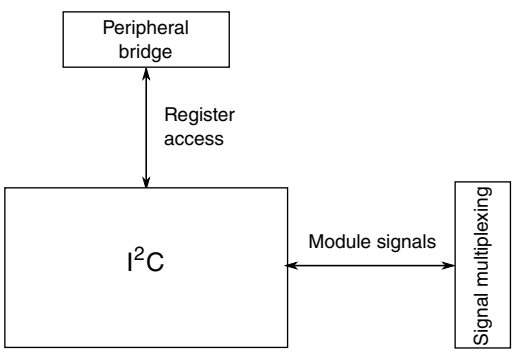


Figure 3-24. I2C configuration

Table 3-39. Reference links to related information

Topic	Related module	Reference
Full description	I^2C	I2C
System memory map		System memory map
Clocking		Clock Distribution

Table continues on the next page...

Table 3-39. Reference links to related information (continued)

Topic	Related module	Reference
Power management		Power management
Signal multiplexing	SIM's GPS registers	Signal multiplexing

3.9.3.1 I2C module address matching to wake the device from stop mode

When the I2C module is configured as slave and the device enters stop mode, the I2C module cannot wake the device from stop mode if the I2C receives two or more non-matching addresses before it receives the matching address.

Options for working around this situation include the following:

- When operating as an I2C slave and in stop mode: Ensure that the external I2C master sends a matching address to wake the slave before it sends any transaction to other I2C slaves. The user must also ensure that the device does not return to stop mode until after all packets to non-matching addresses have been sent.
- Before receiving I2C packets, use a pin interrupt (any pin, whether or not that pin is being used by the active I2C module) to wake the device.

NOTE

If you use the SDA or SCL pin that the active I2C module is using, the device will wake on every I2C transaction on the bus.

- Use wait mode instead of stop mode.

3.9.4 SCI Configuration

This section summarizes how the module has been configured in the chip. For a comprehensive description of the module itself, see the module's dedicated chapter.

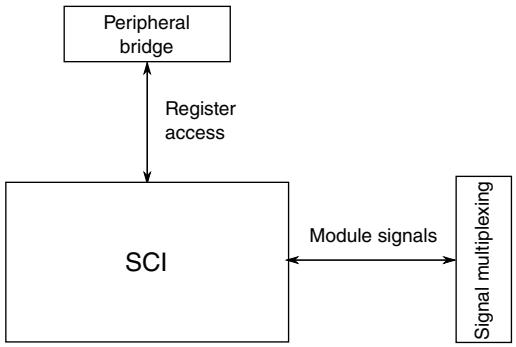


Figure 3-25. SCI Configuration

Table 3-40. Reference links to related information

Topic	Related module	Reference
Full description	SCI	SCI
System memory map		System memory map
Clocking		Clock Distribution
Power management		Power management
Signal multiplexing	SIM's GPS registers	Signal multiplexing

3.10 Human-machine interfaces (HMI)

3.10.1 GPIO Configuration

This section summarizes how the module has been configured in the chip. For a comprehensive description of the module itself, see the module's dedicated chapter.

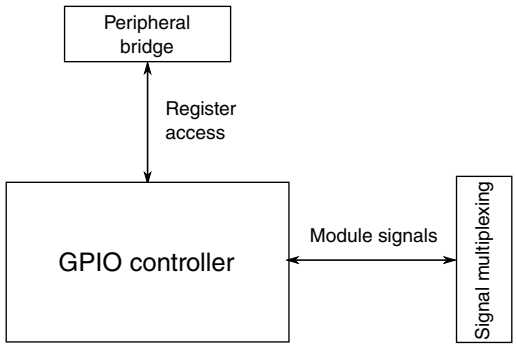


Figure 3-26. GPIO configuration

Table 3-41. Reference links to related information

Topic	Related module	Reference
Full description	GPIO	Parallel Input/Output Control
System memory map		System memory map
Clocking		Clock Distribution
Power management		Power management
Signal multiplexing	SIM's GPS registers	Signal multiplexing

3.10.1.1 GPIO Port D[4:0] configuration

By default, pins 0-4 of GPIO port D are configured differently from other GPIO port pins. These five pins have specific purposes:

- The GPIOD4 pin acts as the RESETB input by default.
- The GPIOD3 pin acts as the TMS input by default.
- The GPIOD2 pin acts as the TCK input by default.
- The GPIOD1 pin acts as the TDO output by default.
- The GPIOD0 pin acts as the TDI input by default.

The following table shows these pins' default configuration at reset. The shaded rows show the values that differ from the reset value of other GPIO port pins.

Table 3-42. GPIOD[4:0] default configuration: register bitfield settings at reset

GPIO Pin	PER[PE]	DRIVE[DRIVE]	SRE[SRE]	PUR[PU]	PUS[PUS]
D4	1	0	1	1	1
D3	1	1	0	1	1
D2	1	0	1	1	0
D1	1	1	0	0	1
D0	1	0	1	1	1

To summarize the default settings for these pins:

- GPIOD[4:0] are configured for peripheral mode.
- GPIOD3 and GPIOD1 are configured for the highest drive strength to enable faster JTAG transactions through the TDO and TMS pins.
- GPIOD[4:2] and GPIO0 each has an internal pulldown resistor enabled.

3.10.1.2 GPIO unbonded pads

Unbonded pads, which are not bonded out to a pin in a particular package, have an automatically disabled pull circuit and input buffer.

Chapter 4
Memory Map

4.1 Introduction

This device contains various memories and memory-mapped peripherals which are located in one contiguous memory space.

4.2 Program/Data Memory Maps

The core processor is word addressed, where a word is 16 bits. Separate memory maps are supported for program space accessed by the Program Data Bus (PDB) and for data space accessed by the primary and secondary data bus (XDB1, XDB2).

NOTE

Both Table 4-1 and Table 4-2 include:

- Primary program/data flash memory
- Secondary (boot) program/data flash memory
- Program/data RAM

Each of these memories consists of a single physical flash memory that is mirrored across the program and data memory maps.

Table 4-1. Program memory map (used by core PDB)

Range	Size (Words)	Use
0x00 0000 - 0x01 ffff	128 KW	Primary program/data flash memory
ox02 0000 - 0x05 ffff	256 KW	Reserved
0x06 0000 - 0x06 3fff	16 KW	Program/data RAM
0x06 4000 - 0x04 7fff	16 KW	Reserved
0x06 8000 - 0x06 bfff	16 KW	Secondary (boot) program/data flash memory
0x06 c000 - 0x06 ffff	16 KW	Reserved

Table continues on the next page...

Table 4-1. Program memory map (used by core PDB) (continued)

Range	Size (Words)	Use
0x07 0000 - 0x07 ffff	64 KW	Reserved
0x08 0000 - 0x1f ffff	1.5 MW	Reserved

Table 4-2. Data memory map (used by core XDB1, XDB2)

Range	Size (Words)	Use
0x00 0000 - 0x00 3fff	16 KW	Program/data RAM
0x00 4000 - 0x00 7fff	16 KW	Reserved
0x00 8000 - 0x00 bfff	16 KW	Secondary (boot) program/data flash
0x00 c000 - 0x00 dfff	8 KW	Core and system peripherals
0x00 e000 - 0x00 ffff	8 KW	Slave peripherals
0x01 0000 - 0x01 dfff	56 KW	Reserved
0x01 e000 - 0x01 ffff	8 KW[1]	FlexRAM
0x02 0000 - 0x03 ffff	128 KW	Primary program/data flash
0x04 0000 - 0x07 ffff	256 KW	Reserved
0x08 0000 - 0xff feff	15.5 MW	Reserved
0xff ff00 - 0xff ffff	128 W	EOnCE registers

1. This device has 1 KW of FlexRAM available when none is configured for use as EEPROM. The upper 7 KW of this space is reserved.

An error results from any attempted access to memory spaces not supported.

The FlexRAM is implemented using the second half of the 32 KB secondary flash memory. An error results from:

- Accessing the second half of secondary flash memory when FlexRAM is enabled as enhanced EEPROM
- Accessing the FlexRAM space beyond its currently configured size

When the FlexRAM is not used as enhanced EEPROM, the FlexRAM space can be accessed as traditional RAM.

4.3 Core and System Peripheral Memory Map

The core and system peripheral memory map is the portion of the data space memory map assigned to core and system peripherals.

Table 4-3. Memory map for core and system peripherals

Peripheral	Instance Name	Base Address (in Words)	Size (in Words)
Core Configuration MCM		0x00 c000	512 w
Reserved		0x00 c200	512 w
Reserved		0x00 c400	512 w
Reserved		0x00 c600	512 w
DMA		0x00 c800	512 w
Reserved		0x00 ca00	512 w
Reserved		0x00 cc00	512 w
Reserved		0x00 ce00	512 w
Reserved		0x00 d000	512 w
Reserved		0x00 d200	512 w
Reserved		0x00 d400	512 w
Reserved		0x00 d600	512 w
Reserved		0x00 d800	512 w
Reserved		0x00 da00	512 w
Reserved		0x00 dc00	512 w
Memory Controllers		0x00 de00	512 w

4.4 Slave Peripheral Memory Map

The slave peripheral memory map is the portion of the data space memory map assigned to slave peripherals.

Table 4-4. Memory map for slave peripherals

Peripheral	Instance Name	Base Address (in Words)	Size (in Words)
12-bit DAC	DAC	0x00 e000	16
Reserved		0x00 e010	16
Comparator (with 6-bit reference DAC)	CMPA	0x00 e020	8
Comparator (with 6-bit reference DAC)	CMPB	0x00 e028	8
Comparator (with 6-bit reference DAC)	CMPC	0x00 e030	8
Comparator (with 6-bit reference DAC)	CMPD	0x00 e038	8
Reserved		0x00 e040	64
Queued Serial Communications Interface module	QSCI0	0x00 e080	16
Queued Serial Communications Interface module	QSCI1	0x00 e090	16
Reserved		0x00 e0a0	16
Queued Serial Peripheral Interface module	QSPI0	0x00 e0b0	16

Table continues on the next page...

Table 4-4. Memory map for slave peripherals (continued)

Peripheral	Instance Name	Base Address (in Words)	Size (in Words)
Reserved		0x00 e0c0	16
Reserved		0x00 e0d0	16
Inter-Integrated Circuit module	I2C0	0x00 e0e0	16
Inter-Integrated Circuit module	I2C1	0x00 e0f0	16
Programmable Interval Timer	PIT0	0x00 e100	16
Programmable Interval Timer	PIT1	0x00 e110	16
Programmable Delay Block	PDB0	0x00 e120	16
Programmable Delay Block	PDB1	0x00 e130	16
General Purpose Timer	TMRA0	0x00 e140	16
General Purpose Timer	TMRA1	0x00 e150	16
General Purpose Timer	TMRA2	0x00 e160	16
General Purpose Timer	TMRA3	0x00 e170	16
General Purpose Timer	TMRB0	0x00 e180	16
General Purpose Timer	TMRB1	0x00 e190	16
General Purpose Timer	TMRB2	0x00 e1a0	16
General Purpose Timer	TMRB3	0x00 e1b0	16
Cyclical Redundancy Check module	CRC	0x00 e1c0	16
Reserved		0x00 e1d0	16
Reserved		0x00 e1e0	32
General Purpose I/O	GPIOA	0x00 e200	16
General Purpose I/O	GPIOB	0x00 e210	16
General Purpose I/O	GPIOC	0x00 e220	16
General Purpose I/O	GPIOD	0x00 e230	16
General Purpose I/O	GPIOE	0x00 e240	16
General Purpose I/O	GPIOF	0x00 e250	16
Reserved		0x00 e260	16
Reserved		0x00 e270	48
Power Management Controller	PMC	0x00 e2a0	16
On-Chip Clock Synthesis module	OCCS	0x00 e2b0	16
Reserved		0x00 e2c0	64
Interrupt Controller	INTC	0x00 e300	32
Computer Operating Properly module	COP	0x00 e320	16
External Watchdog Monitor	EWM	0x00 e330	16
Inter-Peripheral Crossbar Switch	XBARA	0x00 e340	32
Inter-Peripheral Crossbar Switch: AOI Input	XBARB	0x00 e360	32
Crossbar AOI Module	AOI	0x00 e380	32
Reserved		0x00 e3a0	32
Flash Memory interface	FTFL	0x00 e3c0	16
System Integration Module	SIM	0x00 e400	256

Table continues on the next page...

Table 4-4. Memory map for slave peripherals (continued)

Peripheral	Instance Name	Base Address (in Words)	Size (in Words)
12-bit Cyclic ADC	ADC1	0x00 e500	128
16-bit SAR ADC	ADC0	0x00 e580	128
Pulse Width Modulator with nano-edge placement	PWMA	0x00 e600	256
Reserved		0x00 e700	256
FlexCAN3 interface	CAN	0x00 e800	2048
Reserved		0x00 F000	4096

Chapter 5
Clock Distribution

5.1 Overview

Clock generation is performed by two modules, the OCCS and SIM. The OCCS encapsulates all on-chip oscillators and a PLL. Its role is to provide to the SIM a master clock (MSTR_2X) running at two times the system rate. The OCCS's internal oscillators are also output to the COP and EWM for use as secondary clock sources.

The SIM supplies the MSTR_2X clock directly to high-speed (2x) clock applications and divides it by two to generate normal-rate system and peripheral clocks.

The on-chip oscillators internal to the OCCS module are:

- 8 MHz relaxation oscillator
- 32 kHz relaxation oscillator
- 8 MHz to 16 MHz crystal oscillator
- Phase lock loop (PLL)

5.2 Clock definitions

The OCCS is the primary clock generation module and interfaces with the SIM. The SIM outputs system and peripheral clocks that are consumed by the various system-level modules and peripherals for which they are intended.

Clock	Definition
ROSC_8M	Relaxation oscillator 8 MHz clock in the OCCS module.
ROSC_32K	Relaxation oscillator 32 kHz clock in the OCCS module.
XTAL_OSC	Crystal oscillator clock in the OCCS module.

Table continues on the next page...

Clock	Definition
MSTR_2X	Master clock output by the OCCS to the SIM. The SIM uses it to generate clocks to the core and peripherals.
CPU_CLK	System rate clock used by the core: MSTR_2X divided by two. Gated in low power modes and for core stalls.
BUS_CLK	Normal rate peripheral clock: MSTR_2X divided by two. The SIM provides user control of individual peripheral clock operation in wait and stop modes. By default, peripheral clocks are off until they are enabled. Then they operate only in run and wait mode unless they are specifically enabled to run in stop mode as well.
2X_BUS_CLK	Two times the BUS_CLK frequency.
DIV4_BUS_CLK	Clock for memory including flash memory and FlexRAM.
FTFL_OSC	Internal 25 MHz clock of the flash memory (FTFL) module that is used for program/ erase.
TCK	System rate clock used by core for EOnCE operations and enabled only in debug mode.

5.3 System clock source configuration

The following table identifies the possible combinations of clock sources for the system clock. By default, the chip boots with the 8 MHz internal Relaxation Oscillator. After boot, the user can follow the configuration steps in the table to use another clock source.

At the end of each set of configuration steps, the user can change the COD, if desired.

Table 5-1. System clock source configuration

Clock Source	Maximum Frequency	Configuration Steps
8 MHz Relaxation Oscillator (ROSC_8M)	8 MHz	Default
32 kHz Relaxation Oscillator (ROSC_32K)	32 kHz	1. Select the 32 kHz oscillator (PRECS=10b). 2. Wait 6 NOPs for resynchronization. 3. The relaxation oscillator can optionally be powered down (ROPD=1).
External Clock Source	50 MHz	1. Enable the clock source (CLKIN0/CLKIN1) in the GPIO and SIM. 2. Select CLKIN as the source clock (PRECS=01b, EXT_SEL=1). 3. Wait 6 NOPs for the synchronizing circuit to change clocks. 4. The relaxation oscillator can optionally be powered down (ROPD=1).
PLL with internal 8 MHz reference	400 MHz	1. Set PLLDB for desired multiplier and enable the PLL (PLLPD=0). 2. Wait for PLL lock (LCK1=1 and LCK0=1). 3. Change ZSRC to select a PLL-based source.
PLL with CLKIN as reference	400 MHz	1. Enable the clock source (CLKIN) in the GPIO and SIM. 2. Select CLKIN as the source clock (PRECS=01b, EXT_SEL=1). 3. Wait 6 NOPs for the synchronizing circuit to change clocks. 4. The relaxation oscillator can optionally be powered down (ROPD=1). 5. Set PLLDB for desired multiplier and enable the PLL (PLLPD=0). 6. Wait for PLL lock (LCK1=1 and LCK0=1). 7. Change ZSRC to select a PLL-based output clock.

Table continues on the next page...

Table 5-1. System clock source configuration (continued)

Clock Source	Maximum Frequency	Configuration Steps
Crystal Oscillator (XTAL_OSC)	16 MHz	1. Enable the pin functions XTAL and EXTAL in the GPIO and SIM. 2. Change the oscillator to low power mode (COHL=1). 3. Change the external clock source to the crystal oscillator (EXT_SEL=0). 4. Power up the crystal oscillator (CLK_MODE=0). 5. Wait for the oscillator to stabilize (up to 10 ms). 6. Select the crystal oscillator clock source (PRECS=01b). 7. Wait 6 NOPs for the synchronizing circuit to change clocks. 8. The relaxation oscillator can optionally be powered down (ROPD=1).
PLL with Crystal Oscillator as reference	400 MHz	1. Enable the pin functions XTAL and EXTAL in the GPIO and SIM. 2. Change the oscillator to low power mode (COHL=1). 3. Change the external clock source to the crystal oscillator (EXT_SEL=0). 4. Power up the crystal oscillator (CLK_MODE=0). 5. Wait for the oscillator to stabilize (up to 10 ms). 6. The crystal oscillator clock source should be selected (PRECS=01b). 7. Wait 6 NOPs for the synchronizing circuit to change clocks. 8. The relaxation oscillator can optionally be powered down (ROPD=1).

5.4 Module clocks

The following table summarizes the clocks associated with each module.

Table 5-2. Module clocks

Module	Input Clock Option(s)
Core and system modules	
CPU	CPU_CLK
Crossbar switch (AXBS_Lite)	BUS_CLK
Peripheral bridge (AIPS_Lite)	BUS_CLK
XBARs	BUS_CLK
AOI	BUS_CLK
INTC	BUS_CLK
PMC	BUS_CLK, MSTR_CLK
DMA controller	BUS_CLK
EOnCE	TCK
Memories and memory interfaces	
RAM controller	BUS_CLK, 2X_BUS_CLK
System RAM	2X_BUS_CLK
FMC	BUS_CLK
Flash memory (FTFL)	DIV4_BUS_CLK, FTFL_OSC
FlexRAM	DIV4_BUS_CLK, FTFL_OSC

Table continues on the next page...

Table 5-2. Module clocks (continued)

Module	Input Clock Option(s)
FlexCAN RAM	BUS_CLK
Security and integrity modules	
COP	BUS_CLK, XTAL_OSC, ROSC_8M, ROSC_32K
EWM	BUS_CLK, XTAL_OSC, ROSC_8M, ROSC_32K
CRC	BUS_CLK
Analog	
12-bit Cyclic ADC	BUS_CLK, ROSC_8M
16-bit SAR ADC	4MHZ_ROSC (ROSC_8M/2), BUS_CLK, internal ALTCLK
12-bit DAC	BUS_CLK
CMPs (including 6-bit DACs)	BUS_CLK
PWMs and timers	
PWMA with NanoEdge placement	BUS_CLK
PDBs	BUS_CLK
TMRs	BUS_CLK
PITs	BUS_CLK, XTAL_OSC, ROSC_8M, ROSC_32K
Communication interfaces	
SPI	BUS_CLK
SCIs	BUS_CLK, 2X_BUS_CLK
I2Cs	BUS_CLK
FlexCAN3	BUS_CLK, XTAL_OSC, XTAL_OSC_DIV2, CLKIN
CAN_GLITCH_FLT	4MHZ_ROSC (ROSC_8M/2)
Human-machine interface (HMI)	
GPIO	BUS_CLK

Chapter 6
Reset and Boot

6.1 Reset Configuration

Reset is managed by the SIM. The chip supports the following specific sources of reset:

- Power-on reset (POR) from the PMC
- External (PIN) reset from the RESET pin
- COP CPU (processor watchdog) reset from the COP module
- COP loss of clock reference (LOR) reset from the COP module
- Software reset from SIM module

Each peripheral can be individually reset by toggling the corresponding bit in the SIM peripheral reset registers. These registers are write protected.

The power-on reset, by design, supports hysteresis so that it does not release until the supply has risen above the high LVI detection level of 2.7 V. This reset does not reassert until the supply reaches 2.0 V. The PMC is configurable to detect rising or falling transitions through high LVI at 2.7 V or low LVI at 2.2 V so that software can appropriately manage power consumption.

The external pin reset is a selectable pad function of GPIOD4 and is selected at reset. If this pad function is deselected because of the configuration of GPIOD4 or muxing for GPIOD4, the external reset presents a deasserted value to the chip.

The COP provides a mechanism for detecting runaway code on the processor. When enabled, failure of software to regularly reset the COP (the timeout period is configurable) results in the COP asserting a reset to the SIM.

The COP also receives a loss-of-reference input from the OCCS module that is associated with its loss-of-reference interrupt. The COP provides a counter of configurable duration. Failure to clear the LOR condition before the timer period elapses results in a loss of reference reset.

In addition to the peripheral bus clock, the COP has access to alternate clock sources, including the 8 MHz / 400 kHz ROSC, 32 kHz ROSC, and crystal oscillator. At any given time, when the OCCS is using a particular clock source to produce the master system clock, configure the COP to use a different clock source to prevent compromised COP functionality in the event of a loss of the system clock.

The software reset provides a capability for user software to reset the entire chip.

6.2 System Boot

The chip has two options for system boot:

- During normal operation, the CPU always boots from internal flash memory. There is no external memory interface.
- During development, the core can boot directly into debug mode.

6.2.1 FOPT boot options

The flash option (FOPT) register in the flash memory (FTFL) module allows the user to customize the operation of the core at boot time. The register contains read-only bits that are loaded from the NVM's option byte in the flash configuration field. The user can reprogram the option byte in flash to change the FOPT values that are used for subsequent resets. For more details on programming the option byte (OPT field), refer to the detailed description of the flash memory module.

The core uses the FTFL_FOPT register bits to configure the device at reset as shown in the following table.

Table 6-1. Flash Option Register (FTFL_FOPT) Bit Definitions

Bit Number	Field	Value	Definition
7-1	Reserved		Reserved for future use.
0	Advanced Low Power Mode Enable	0	At reset exit, the chip's power modes are controlled through the SIM_PWR register (the SIM_PWRMODE register has no effect).
		1	At reset exit, the chip's power modes—including advanced low power modes—are controlled through the SIM_PWRMODE register (the SIM_PWR register has no effect).

6.2.2 Boot Procedure for Normal Operation

The chip has a number of reset sources:

1. Internal power-on reset
2. External RESET pin
3. Software reset
4. COP CPU reset
5. COP loss of clock reference reset

Assuming that the JTAG port is in its reset state, the first three of these cause the CPU to boot from the locations at 00_0000h. Both COP resets cause the CPU to boot from the locations at 00_0002h.

6.2.3 Boot sequence

At power up, the Relaxation Oscillator (ROSC) starts to operate. The on-chip regulator holds the system in a POR state until the input supply is above the POR threshold. The system continues to be held in this static state until the input supply reaches a safe operating voltage as determined by the LVD. After the input supply crosses this LVD threshold, the Power Management Controller releases the POR signal to the system. After POR deassertion, the SIM starts the reset exit sequence.

1. A system reset is held on internal logic, and the OCCS module is enabled in its default clocking mode. The system begins to receive the clock from the internal ROSC.
2. Required clocks are enabled: core clock, system clock, flash clock, and any bus clocks that do not have clock gate control.
3. The POR is extended for 64 ROSC clock cycles. The deassertion of this extended POR enables the clock dividers within the SIM (for the bus clock and flash clock).
4. The system reset on internal logic continues to be held. However, the Flash Memory Controller is released from reset (by early reset deassertion) and begins initialization operation, which includes the following events:
   - The FTFL_FSEC register is updated and the security state is established.
   - NVOPT and IFR information is delivered from the flash and gets captured in SIM.
5. If $\overline{\text{RESET}}$ continues to be asserted (an indication of a slow rise time on the $\overline{\text{RESET}}$ pin or external drive in low), the system continues to be held in reset. After the $\overline{\text{RESET}}$ pin is detected to be high and flash initialization completes, the system is released from reset.
6. When the system exits reset, the processor fetches initial 16-bit values from the location specified by the Vector Address Base of the Interrupt Controller. By default, the location is 00_0000h, which is the location of program flash memory, but the location is 00_0002h in case of COP reset sources.

- If FlexNVM is enabled, then the Flash Memory Controller continues to restore the FlexNVM data. This data is not available immediately out of reset, and the system should not access this data until the Flash Memory Controller completes this initialization step, which is indicated by the EEERDY flag.

Subsequent system resets follow this reset flow beginning with the step where system clocks are enabled.

Chapter 7
Power Management

7.1 Overview

Power is supplied through external pads for digital and analog power and ground. The analog supply VDDA is consumed directly by analog components. The external digital supply VDD (externally in common with VDDA) is regulated to produce stable voltages for internal digital applications. Safe operation is insured by power-on reset and low voltage detection circuits which allow the user to manage power consumption due to insufficient operating voltage.

Power consumption during part operation is managed by powering off non-essential analog components and through the use of low-power modes (RUN, WAIT, STOP) and clock gating associated with these modes.

7.2 Architecture

Internal rails are generated from external VDD by a large and small power regulator in the PMC module. The small regulator provides isolated power for clock generation and other noise-critical applications. The large regulator supplies all other internal digital applications.

The PMC incorporates power-on reset logic and two levels of low voltage detection. A hysteresis circuit ensures that power-on reset does not release until the supply rises above the LV1 threshold where the part can operate safely over PVT. On falling voltage conditions, the part continues to operate below LV1 and a lower LV2 until a POR level is reached where the part is ultimately forced to reset. This approach offers the opportunity to disable nonessential operations and configure the part in a safe minimum power state until power is restored.

The primary methods for power management during part operation are module-specific clock enables, clock frequency control, clock gating associated with the use of low power modes, and clock gating added by power synthesis. Module enables are generated by modules to turn off generation of their clocks when they are not required. The OCCS module provides flexible clock frequency control. Clock gating and low power modes are implemented by the processor and SIM and cause clocks to be generated only when enabled for operation in the current power mode. Clock tree synthesis further limits power by denying clocks to registers that are not changing state.

7.3 External Supplies and Regulation

Power regulation is managed by the Power Management Controller. The PMC uses a small regulator to maintain a stable, low-noise 2.7 V supply to the PMC, oscillators, PWMA with NanoEdge placement, and PLL. It also uses the small regulator to maintain a stable and relatively low-noise 1.2 V supply to the PLL and PWMA with NanoEdge placement. The PMC uses a large regulator to maintain a stable and relatively low-noise 1.2 V supply to other internal digital circuitry.

As voltage increases at power on, POR does not release until a safe operating voltage of 2.7 V is achieved on the external digital rail. This value ensures that the internal regulators are operating at their intended threshold before POR is released.

Should the external digital supply (VDD) fall, the PMC detects and can generate interrupts for an LVI1 threshold violation at 2.7 V and an LVI2 threshold violation at 2.2 V. Interrupts associated with these thresholds can be used for disabling nonessential features and eventually disabling applications until power is restored or the part is reset. In the case of falling voltage, the POR does not assert until the low POR threshold is violated at 2.0 V, so the maximum time is allowed for orderly shutdown.

7.4 User Power Management Methods

The primary means available to the user to manage power consumption are clock enables, limiting clock frequency, using low power modes, and using DMA. Further indirect power savings can be achieved by reducing unnecessary signal transitions through careful crafting of application software.

The most basic form of power control are module enables. Selected peripherals provide their own clock gating controls back to the SIM. Using these controls, clock generation is gated off automatically in the SIM when not required by the peripheral. Module enables are also used to power off embedded analog functions when not in use. It is therefore advantageous to leave system and peripheral functions disabled if not currently in use.

Clock frequency control is performed in the OCCS. The OCCS provides the means to perform glitch-free transitions between the available clock sources including 8 MHz crystal oscillator, 8 MHz relaxation oscillator, 32 kHz internal oscillator and an external clock. Only clock sources currently in use should be powered on. The PLL should only be powered on when necessary to produce higher operating frequencies. A post scaler with up to 50x division is provided to reduce the active clock source frequency (PLL output or oscillator) even further for power savings. This gives the part flexible frequency control from full speed (100MHz system bus rate) to below 1 kHz.

Low power modes are entered via the processor upon execution of STOP or WAIT instructions and conveyed to the SIM, which performs related clock gating for RUN, WAIT, and STOP modes. The processor and processor dependent clocks operate only in RUN mode. Peripheral clocks do not operate in any power mode until enabled individually by the user in the SIM. Once enabled, they normally run only in RUN and WAIT modes and must be specifically overridden to remain clocked in STOP mode.

The DMA controller contributes to power management by permitting peripheral I/O to be serviced by the DMA while the core remains unclocked in WAIT mode. By eliminating the requirement for the core to service the related interrupts, the DMA controller decreases the frequency and duration of intervals where the core must be returned to RUN mode to service interrupts.

Additional power savings can be achieved by crafting application software to avoid unnecessary register state transitions and to configure module-specific frequency controls to use the lowest-supportable operating rates that satisfy application requirements.

7.5 Power Modes

The CPU has three primary modes of operation: run, wait, and stop mode. Additional low power operating modes can reduce run-time power when maximum bus frequency is not required to support application needs.

The large regulator can support both full power and low power mode (also called the large regulator's standby mode). The maximum operating frequency in this low power mode is 2 MHz.

The small regulator can support full power, low power, and power down modes. In the small regulator's power down mode, the 2.7 V supply is completely shut off. As a result, all on-chip clock sources are disabled. To operate the chip in this mode, the user must provide an external clock through the CLKIN0/1 pin.

The flash memory (FTFL) module has its own power modes: full power mode, Very Low Power (VLP) mode, and Very Low Power Stop (VLPS) mode.

DMA can be enabled in any of the power modes.

Various of the low power modes function only if the FOPT[0] bit is set.

Table 7-1. Power Modes of Operation

Chip Power Mode	CPU Clock	Peripheral Clock	Small Regulator 2.7 V	Large/Small Regulator 1.2 V	Flash Power Mode	Max. System Clock Freq.	Wakeup Source
RUN	ON	ON	Full Power	Full Power	Full Power	100 MHz	NA
WAIT	OFF	ON	Full Power	Full Power	Full Power	100 MHz	Peripheral interrupt and/or reset
STOP	OFF	OFF[2]	Full Power	Full Power	Full Power	100 MHz	Interrupt from clock enabled peripherals (SD bit support)/ reset/ async interrupts (I2C, SCI, CMP, SAR ADC, FlexCAN3)
LPRUN	ON	ON	Low Power Mode	Low Power Mode	VLP[3]	2 MHz	Clear the standby bit in SIM_PWR register.
LPWAIT	OFF	ON	Low Power Mode	Low Power Mode	VLP[3]	2 MHz[3]	Peripheral interrupt and/or reset
LPSTOP[1]	OFF	OFF[2]	Low Power Mode	Low Power Mode	VLPS[3]	2 MHz[2, 3]	Interrupt from clock enabled peripherals (SD bit support)/ reset/ async interrupts (I2C, SCI, CMP, SAR ADC , FlexCAN3)
VLPRUN[4]	ON	ON	Power Down Mode	Low Power Mode	VLP	200 kHz	Clear the Power Down bit and

Table continues on the next page...

Table 7-1. Power Modes of Operation (continued)

Chip Power Mode	CPU Clock	Peripheral Clock	Small Regulator 2.7 V	Large/Small Regulator 1.2 V	Flash Power Mode	Max. System Clock Freq.	Wakeup Source
							STDBY bit in SIM_PWR register.
VLPWAIT	OFF	ON	Power Down Mode	Low Power Mode	VLP	200 kHz	Peripheral interrupt and/or reset
VLPSTOP[1, 4]	OFF	OFF[2]	Power Down Mode	Low Power Mode	VLPS	200 kHz[2]	Interrupt from clock enabled peripherals (SD bit support)/ reset/ async interrupts (I2C, SCI, CMP, SAR ADC, FlexCAN3)

1. If DMA is enabled in any STOP mode, flash memory will not enter its VLPS mode.
2. In all STOP modes, the clock to some portions of SIM logic is never gated. In addition, the user can enable the clock of select peripherals by setting the corresponding bit in one of the SIM's SDn registers.
3. For any chip LP mode, the clock source is the ROSC or XOSC. In that case, the maximum system frequency is 2 MHz. The PLL is shut down in LP modes. In all chip LP modes and flash memory VLP modes, the maximum frequency for flash memory operation is 500 kHz due to the fixed frequency ratio of 1:4 between the CPU clock and the flash clock.
4. When the chip is in any VLP mode, all internal clock sources are unavailable. To operate the chip in those modes, the user must provide a clock through the CLKIN0/1 port.

7.6 Power mode transitions

The following figure shows the chip's power modes and the available transitions among them.

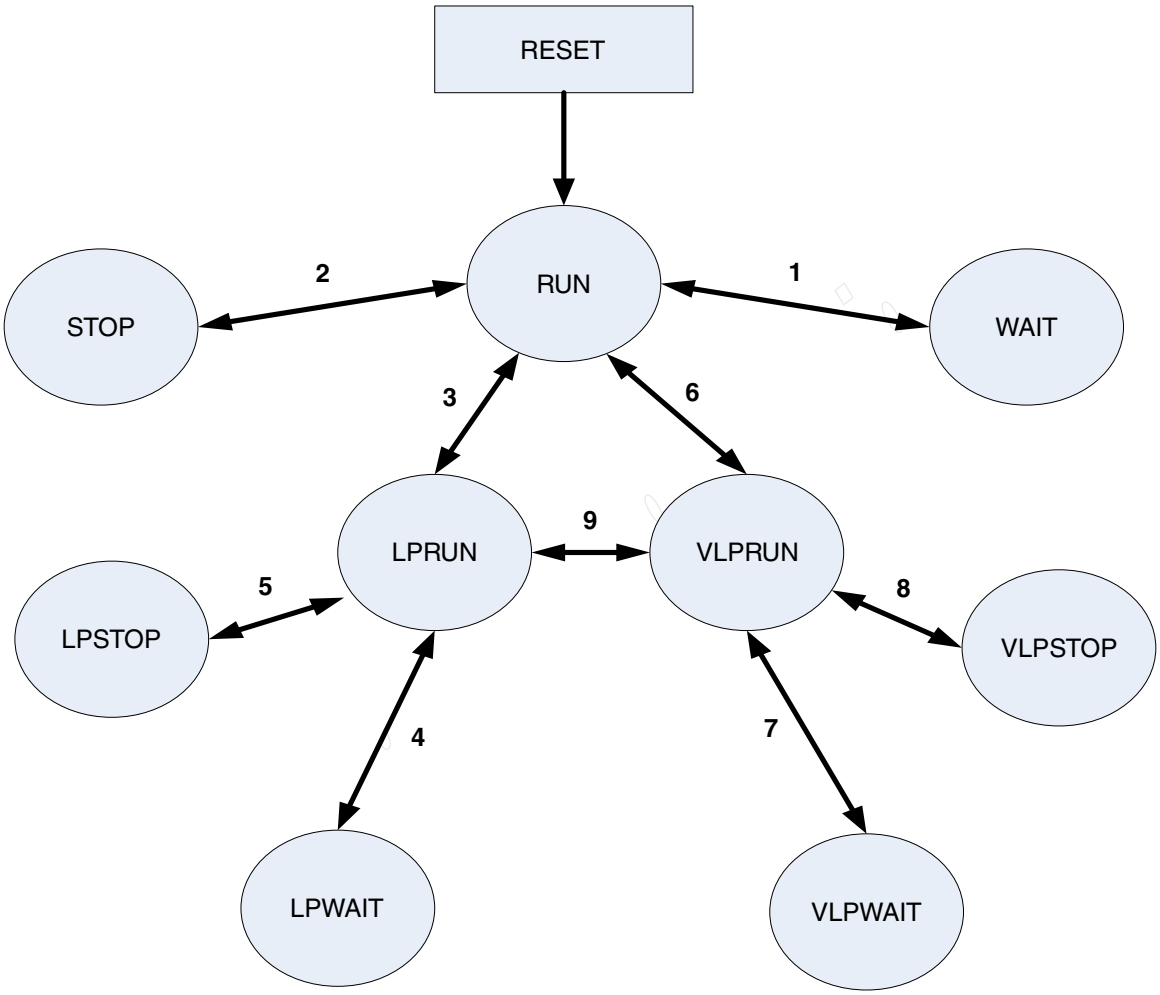


Figure 7-1. Power mode state transitions

The following table defines triggers for the various state transitions shown in the preceding figure.

NOTE

To prevent current leakage in any VLP mode, ensure the following settings apply before entering the VLP mode:

1. The OCCS_CTRL[PLLPD] bit is 1.
2. The PWMA_FRCTRL[FRAC_PU] bit is 0.
3. The OCCS_OSCTL1[ROPD], OSCTL2[COPD], and OSCTL2[ROPD32K] bits are each 1.

Table 7-2. Power mode transitions

Transition number	From	To	Device configuration and/or trigger condition
1	RUN	WAIT	1. Ensure the SIM_CTRL[0] bit is clear. 2. Execute the WAIT instruction.

Table continues on the next page...

Table 7-2. Power mode transitions (continued)

Transition number	From	To	Device configuration and/or trigger condition
	WAIT	RUN	Synchronous interrupt or reset
2	RUN	STOP	1. Ensure SIM_CTRL[2] is clear. 2. Execute the STOP instruction.
	STOP	RUN	1. Synchronous interrupt from a peripheral whose clock is enabled in stop mode by setting the corresponding SIM_SDn bit. 2. Asynchronous interrupt from an I2C module, the FlexCAN module, a CMP module, the SAR ADC module, or an LVD event. 3. POR or PIN reset.
3	RUN	LPRUN	1. If the system is running with the PLL, then switch the clock source to XOSC/ROSC by setting OCCS_CTRL[ZRSR] to 01b. 2. Before changing ZSRC, ensure the MSTR_OSC clock is stable. 3. Put the PLL in power down mode by setting OCCS_CTRL[PLLPD]. 4. If the internal oscillator is used during LP mode, put the crystal oscillator in power down mode by setting OSCTL2[COPD] to 1. 5. Similarly, if the crystal oscillator is used, put the ROSC in power down mode by setting OSCTL1[ROPD] to 1 and OSCTL2[ROPD32K] to 1. 6. Configure the OCCS_DIVBY[COD] field to ensure the system clock frequency (2x) does not exceed 4 MHz. 7. Set SIM_PMODE[LPMODE].
	LPRUN	RUN	Clear the SIM_PWRMODE[LPMODE] bit.
4	LPRUN	LPWAIT	1. Ensure SIM_CTRL[0] is clear. 2. Execute the WAIT instruction.
	LPWAIT	LPRUN	Synchronous interrupt or reset
5	LPRUN	LPSTOP	1. Ensure SIM_CTRL[2] is clear. 2. Execute the STOP instruction.
	LPSTOP	LPRUN	1. Synchronous interrupt from a peripheral whose clock is enabled in stop mode by setting the corresponding SIM_SDn bit. 2. Asynchronous interrupt from an I2C module, the FlexCAN module, a CMP module, the SAR ADC module, or an LVD event. 3. POR or PIN reset.
6	RUN	VLPRUN	1. Configure OCCS registers (EXT_SEL, PRECS, ZSRC, and COD) to ensure the CLKIN path is selected as the system clock (2x) with a maximum frequency of 4 MHz. 2. Set SIM_PWRMODE[VLPMODE] to 1.
	VLPRUN	RUN	1. Clear the SIM_PWRMODE[VLPMODE] and SIM_PWRMODE[LPMODE] bits (if they are already set). 2. Wait for the PMC[SR27] bit to be set.
7	VLPRUN	VLPWAIT	1. Ensure SIM_CTRL[0] is clear. 2. Execute the WAIT instruction.
	VLPWAIT	VLPRUN	Synchronous interrupt or reset
8	VLPRUN	VLPSTOP	1. Ensure SIM_CTRL[2] is clear. 2. Execute the STOP instruction.
	VLPSTOP	VLPRUN	1. Synchronous interrupt from a peripheral whose clock is enabled in stop mode by setting the corresponding SIM_SDn bit. 2. Asynchronous interrupt from an I2C module, the FlexCAN module, a CMP module, the SAR ADC module, or an LVD event. 3. POR or PIN reset.

Table continues on the next page...

Table 7-2. Power mode transitions (continued)

Transition number	From	To	Device configuration and/or trigger condition
9	LPRUN	VLPRUN	1. Configure OCCS registers (EXT_SEL, PRECS, ZSRC, and COD) to ensure the CLKIN path is selected as the system clock (2x) with a maximum frequency of 4 MHz. 2. Set SIM_PWRMODE[VLPMODE] to 1.
	VLPRUN	LPRUN	1. Ensure the LPMODE bit is set. 2. Clear SIM_PWRMODE[VLPMODE] to 0. 3. Wait for PMC[SR27] bit to be set.

Chapter 8
Signal Multiplexing and Signal Descriptions

8.1 Introduction

To optimize functionality in small packages, pins have several functions available via signal multiplexing. This chapter illustrates which of this device's signals are multiplexed on which external pin.

8.2 Signal Multiplexing Integration

This information summarizes how the functionality is integrated into the chip. For more information, see the the following references.

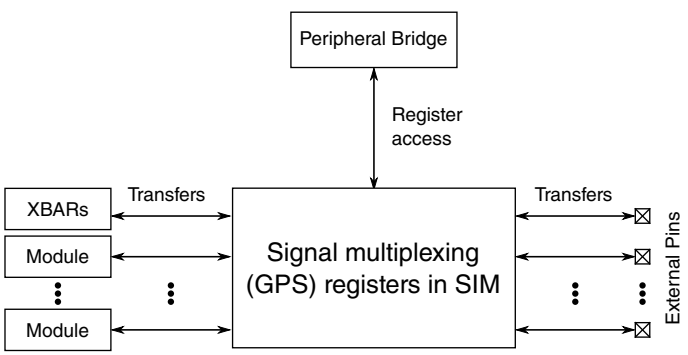


Figure 8-1. Signal multiplexing integration

Table 8-1. Reference links to related information

Topic	Related module	Reference
Full description	SIM	System Integration Module (SIM)
System memory map		System Memory Map
Clocking		Clock Distribution
Register access	Peripheral Bridge	

8.3 Signal Multiplexing and Pin Assignments

This section shows the signals available on each package pin and the locations of these pins on the devices supported by this document. The SIM's GPS registers are responsible for selecting which ALT functionality is available on most pins.

NOTE

The RESETB pin is a 3.3 V pin only.

NOTE

If the GPIOC1 pin is used as GPIO, the XOSC should be powered down.

64 LQFP	Default	ALT0	ALT1	ALT2	ALT3
1	TCK	GPIOD2			
2	RESETB	GPIOD4			
3	GPIOC0	EXTAL	CLKIN0		
4	GPIOC1	XTAL			
5	GPIOC2	TXD0	TB0	XB_IN2	CLKO0
6	GPIOF8	RXD0	TB1	CMPD_O	
7	GPIOC3	TA0	CMPA_O	RXD0	CLKIN1
8	GPIOC4	TA1	CMPB_O	XB_IN8	EWM_OUT_B
9	GPIOA7	ANA7&ANC11			
10	GPIOA6	ANA6&ANC10			
11	GPIOA5	ANA5&ANC9			
12	GPIOA4	ANA4&ANC8&CMPD_IN0			
13	GPIOA0	ANA0&CMPA_IN3	CMPC_O		
14	GPIOA1	ANA1&CMPA_IN0			
15	GPIOA2	ANA2&VREFHA&CMPA_IN1			
16	GPIOA3	ANA3&VREFLA&CMPA_IN2			
17	GPIOB7	ANB7&ANC15&CMPB_IN2			
18	GPIOC5	DACO	XB_IN7		
19	GPIOB6	ANB6&ANC14&CMPB_IN1			
20	GPIOB5	ANB5&ANC13&CMPC_IN2			
21	GPIOB4	ANB4&ANC12&CMPC_IN1			
22	VDDA				
23	VSSA				
24	GPIOB0	ANB0&CMPB_IN3			
25	GPIOB1	ANB1&CMPB_IN0			
26	VCAP				
27	GPIOB2	ANB2&VREFHB&CMPC_IN3			

64 LQFP	Default	ALT0	ALT1	ALT2	ALT3
28	GPIOB3	ANB3&VREFLB&CMPC_IN0			
29	VDD				
30	VSS				
31	GPIOC6	TA2	XB_IN3	CMP_REF	
32	GPIOC7	SS0_B	TXD0		
33	GPIOC8	MISO0	RXD0	XB_IN9	
34	GPIOC9	SCLK0	XB_IN4		
35	GPIOC10	MOSI0	XB_IN5	MISO0	
36	GPIOF0	XB_IN6	TB2	SCLK1	
37	GPIOC11	CANTX	SCL1	TXD1	
38	GPIOC12	CANRX	SDA1	RXD1	
39	GPIOF2	SCL1	XB_OUT6		
40	GPIOF3	SDA1	XB_OUT7		
41	GPIOF4	TXD1	XB_OUT8		
42	GPIOF5	RXD1	XB_OUT9		
43	VSS				
44	VDD				
45	GPIOE0	PWMA_0B			
46	GPIOE1	PWMA_0A			
47	GPIOE2	PWMA_1B			
48	GPIOE3	PWMA_1A			
49	GPIOC13	TA3	XB_IN6	EWM_OUT_B	
50	GPIOF1	CLKO1	XB_IN7	CMPD_O	
51	GPIOE4	PWMA_2B	XB_IN2		
52	GPIOE5	PWMA_2A	XB_IN3		
53	GPIOE6	PWMA_3B	XB_IN4		
54	GPIOE7	PWMA_3A	XB_IN5		
55	GPIOC14	SDA0	XB_OUT4		
56	GPIOC15	SCL0	XB_OUT5		
57	VCAP				
58	GPIOF6	TB2	PWMA_3X		XB_IN2
59	GPIOF7	TB3	CMPC_O	SS1_B	XB_IN3
60	VDD				
61	VSS				
62	TDO	GPIOD1			
63	TMS	GPIOD3			
64	TDI	GPIOD0			

8.4 Pinout diagrams

The following diagrams show pinouts for the packages. For each pin, the diagrams show the default function. However, many signals may be multiplexed onto a single pin.

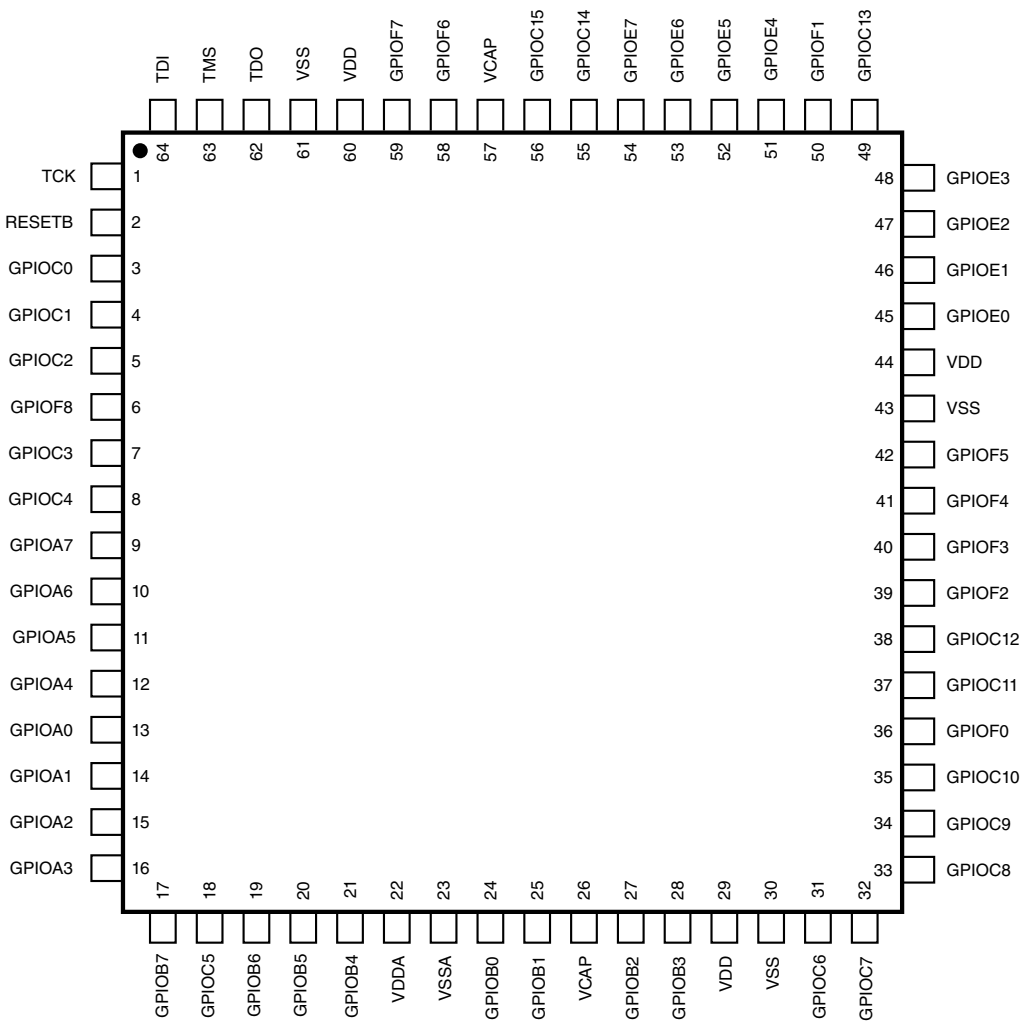


Figure 8-2. 64-pin LQFP

NOTE

The RESETB pin is a 3.3 V pin only.

Chapter 9
Memory Resource Protection (MRP)

Memory resource protection allows two levels of software to co-exist in the core, while maintaining hardware-enforced isolation. The levels are designated supervisor software and user software. The hardware feature along with appropriate software architecture allow the system to protect supervisor programs and resources being accessed from user programs.

This resource protection allows unverified or third-party software to safely run in user mode, allowing it to only access unprotected memory locations and resources.

The software enablement of resource protection is provided in the MCM's Resource Protection Control Register (RPCR).

The resource protection features provided are:

- Partition software into two modes: supervisor software and user software
- Partition system address spaces and resources, both code and data, into two modes
- Provide secure methods of transfer between modes
- Provide separate stacks and stack pointers for supervisor and user modes

9.1 Overview

The software is able to be partitioned into two modes: supervisor and user. The system resources are partitioned for both code and data into supervisor and user regions. Supervisor code can only be executed from supervisor regions and user code can only be executed from user regions. The data regions defined as supervisor can be accessed only by supervisor code; the user data regions can be accessed by both supervisor and user code.

Table 9-1. Resource Protection Accesses

System Resource	Supervisor Code	User Code
Supervisor data	Allowed	Prohibited

Table continues on the next page...

Table 9-1. Resource Protection Accesses (continued)

System Resource	Supervisor Code	User Code
User data	Allowed	Allowed
Peripheral space	Allowed	Prohibited
FlexNVM/FlexRAM	Allowed	Prohibited
Debug resources	Allowed	Prohibited

NOTE

User mode cannot directly access FlexNVM, FlexRAM, peripheral space, or EOnCE registers. It can access only the user mode portion of flash memory and RAM.

Supervisor code may access anywhere in supervisor or user space. Supervisor code may branch into user code by executing one of the three core "return from interrupt" instructions (RTI, RTID, FRTID). User code may access anywhere in user space. If user code attempts to branch into supervisor space or to reference data in supervisor space, the memory reference is aborted, producing a fault and generating an interrupt. User code may safely return control to supervisor program and data memory spaces with system calls using the SWI (software interrupt) instruction. Additionally, any interrupt exception forces entry to supervisor mode.

9.2 Features

The resource protection features are:

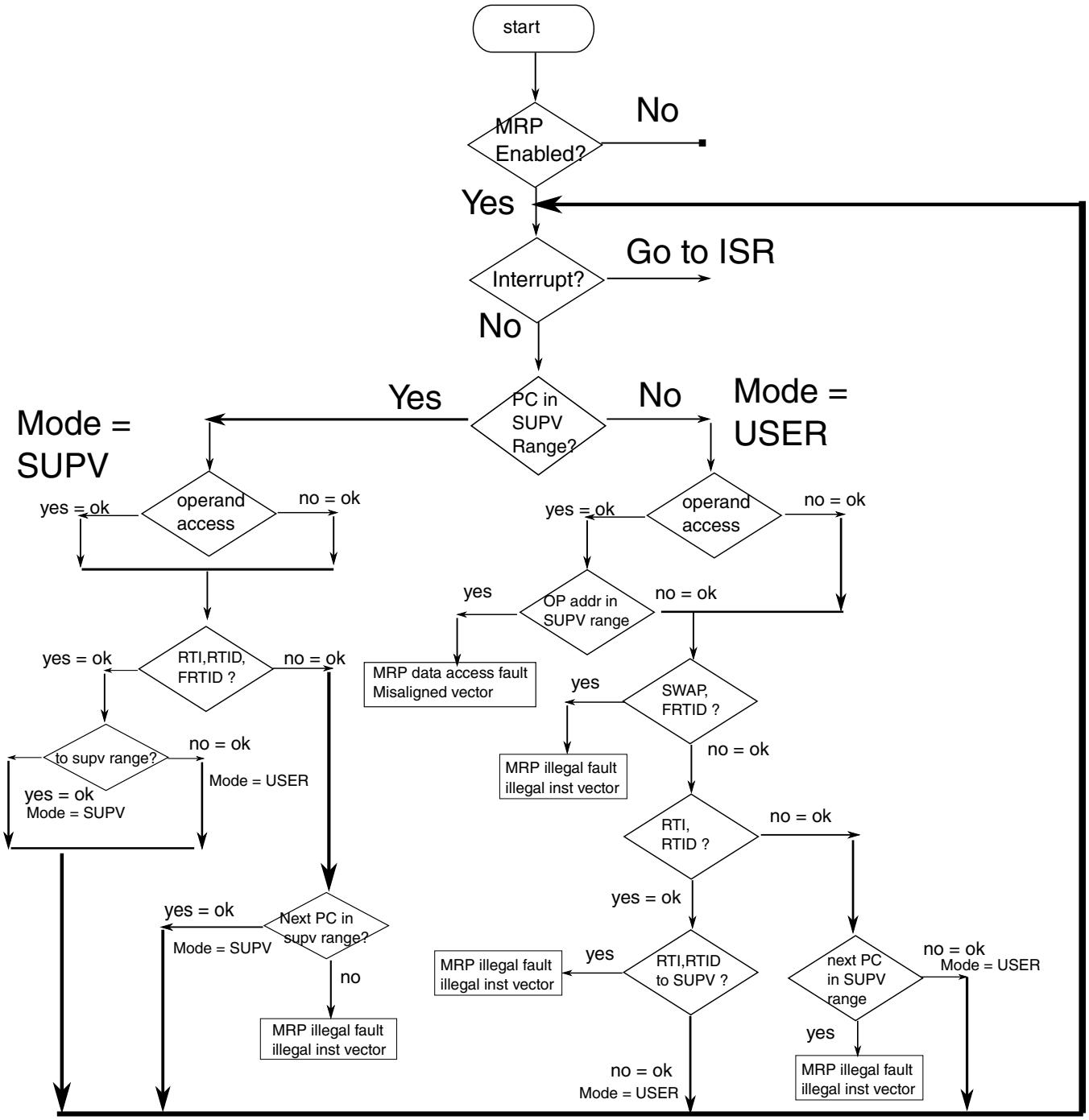
- Two stack pointer registers: active and "other"
- Flash memory regions defined with 8 KB granularity
- RAM memory regions defined with 512 byte granularity
- Hardware-managed stack pointer register visible to software
- Supervisor stack pointer guaranteed on software faults and interrupts
- Stack pointer swap managed in hardware for supervisor-to-user transition via the RTI, RTID, FRTID instructions
- User-to-supervisor transition and stack pointer register swap managed on SWI instruction and all interrupts

9.3 Operation

Each of the protected code and data memory regions—flash memory and RAM—are divided into two address spaces through supervisor-only memory-mapped programmable registers: the UFLASHBAR (User Flash Base Address Register) and UPRAMBAR (User

PRAM Base Address Register) . These registers define the base address for each region. Locations above this base address consist of user space, and locations below the base address are in the supervisor area.

The following diagram shows the allowed transitions between supervisor and user modes. Faulting transitions and illegal data references show the fault indicators for each illegal operation.



Terms:
ifault: Resource Protection illegal fault uses illegal instruction vector
dfault: Resource Protection data access fault uses misaligned vector
SUPV = supervisor
USER = user
operand access: current instruction that performs any operand read or write

Figure 9-1. MRP flows

The transition from supervisor code to user code is managed via the RTI, RTID, and FRTID instructions. The only transition from user code to supervisor code is via an interrupt or a fault triggered by a user software attempt to access a restricted region. Any other attempt results in a fault and subsequent return to supervisor mode. The preferred method for a user program to return to supervisor mode is to perform a system call via the SWI instruction. Additionally, user mode code is restricted from modifying SR (Status Register) bits [9:8] (interrupt mask level bits). Any attempt by user code to modify these bits is ignored.

Region transition is managed via the stack pointer. The core controls the stack pointers (SP) as follows:

1. The hardware manages two SPs: a supervisor SP and a user SP. For correct and secure operation, the supervisor stack is located in supervisor RAM space and the user stack is located in user RAM space. Both spaces are defined by the contents of the UPRAMBAR.
2. The hardware manages one SP as the "active" stack pointer and the alternate as the "other" stack pointer. The active stack pointer resides in the core's SP register. The other stack pointer resides in a supervisor-only memory-mapped register.
3. On all faults and interrupts (supervisor mode entry points), the hardware guarantees the supervisor stack pointer is the active stack pointer. In other words, on all faults and interrupts:
   - If the supervisor stack pointer is in the SP register, interrupt processing continues.
   - If the user stack pointer is the active pointer and in the SP register, the hardware first swaps the SP register with the "other" stack pointer register, activating the supervisor stack pointer and saving the user SP before proceeding with interrupt processing.
4. Similar operations involving the two stack pointers are performed on possible supervisor mode exit points. On a return from interrupt instructions (RTI, RTID, FRTID), if the instruction is in supervisor code space and the target instruction address of the return is in user code space, the hardware:
   a. Swaps the SP holding the supervisor stack pointer with the "other" stack pointer register, activating the user stack pointer and saving the supervisor SP in the “other” SP register.
   b. Passes control to the user mode code.

   If the return target is located in the supervisor address space, then no stack pointer exchange is required and control immediately passes to the target instruction.
5. Stack pointer swaps occur such that all faults and interrupts are processed on the supervisor stack.
6. Stack pointer swaps do not incur any delay (they occur with zero cycle overhead).

The transfer from supervisor to user mode is done by the execution of a RTI, RTID, or FRTID instruction in supervisor space with a target in user space. All other transfers from supervisor code space to user code space result in a fault.

The only allowable state transition from user code to supervisor code is via a fault or an interrupt. The preferred, graceful method for a user program to perform a system call to pass control to supervisor code is via the SWI instruction.

9.4 Programming Model Overview

The resource protection registers are available only when memory resource protection is enabled. These registers reside in the MCM, and the addresses are offsets of the MCM's base address. The MCM is a supervisor-only space, so all registers must be accessed by supervisor code.

9.5 Memory Resource Protection Restrictions

In resource protection mode, the following restrictions apply:

- An attempt to execute the following supervisor-only instructions when in user mode produces an illegal fault:
  - FTRID
  - SWAP
- An attempt to execute the following instructions when in user mode results in an illegal fault if the target is in supervisor space:
  - RTI
  - RTID
- In resource protection mode, the following registers are supervisor only:
  - All shadow registers
- The SR (status register) bits [9:8] (interrupt mask level bits) cannot be modified by user code.

9.6 Base Address Setup

The UFLASHBAR and UPRAMBAR registers express the size of the portion of the flash memory or program RAM that is used for supervisor space when resource protection is enabled. The hardware manages this information correctly for accesses to flash memory or RAM for both program and data memory accesses.

UFLASHBAR Example

If resource protection is enabled and the primary program/data flash memory size is 64 KB (32 KW), setting the UFLASHBAR to 8000h defines the supervisor access region to be 32 KB. As a result, for both the program memory map and the data memory map, the supervisor access region occupies the bottom half of the primary program/data flash memory space and the user access region occupies the upper 32 KB of the space.

- Program (PDB bus) accesses to flash memory use the program memory map. Address accesses in the range of 00_0000h to 00_3FFFh in terms of words are supervisor mode accesses to the primary flash memory. Accesses in the range of 00_4000h to 00_7FFFh in terms of words are user mode accesses to the primary flash memory.
- Data (XAB1 or XAB2 bus) accesses to flash memory use the data memory map. Address accesses in the range of 02_0000h to 02_3FFFh in terms of words are supervisor mode accesses to the primary flash memory. Accesses in the range of 02_4000h to 02_7FFFh in terms of words are user mode accesses to the primary flash memory.

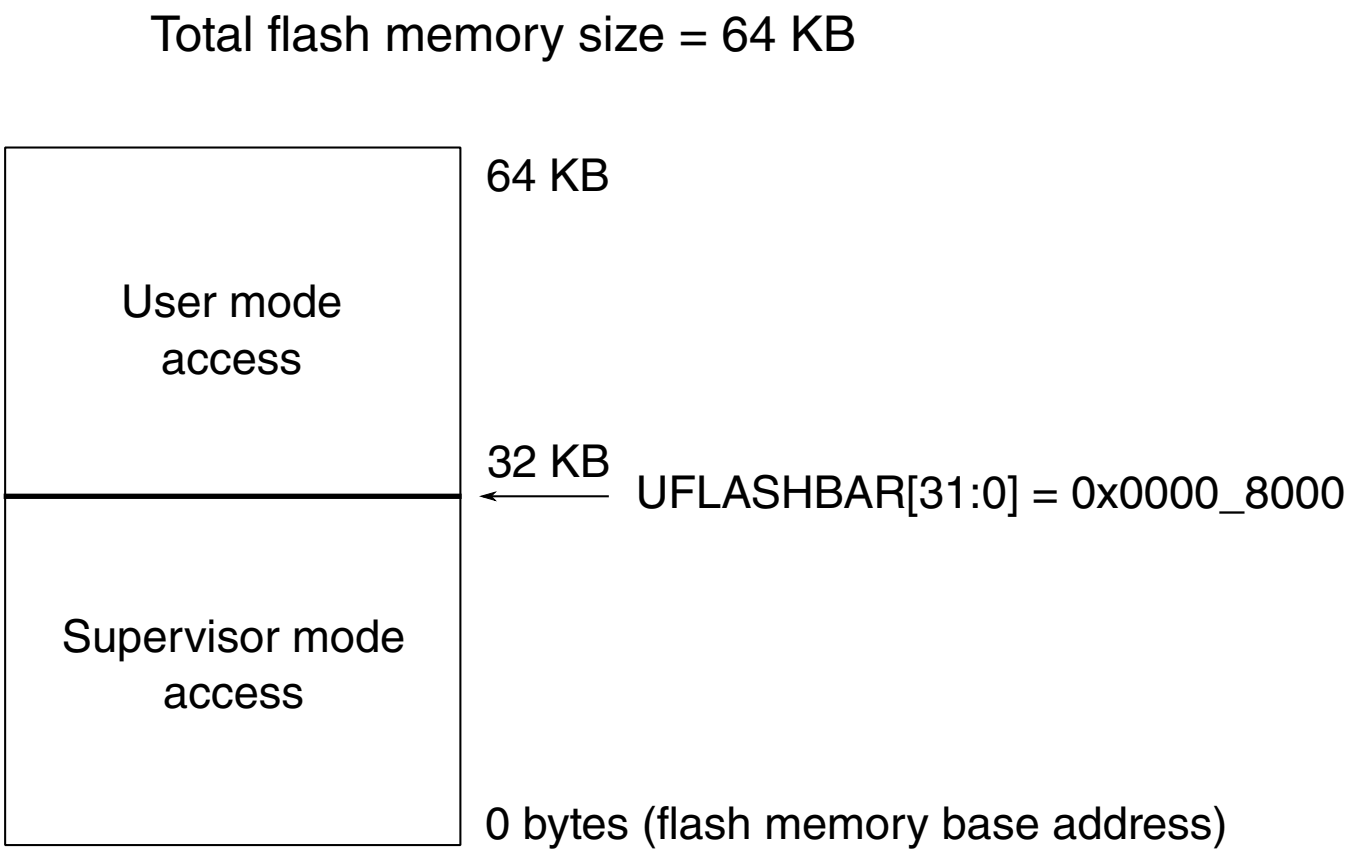


Figure 9-2. Flash Memory Base Address Setup Example

UPRAMBAR Example

If resource protection is enabled and the RAM size is 32 KB (16 KW), setting the UPRAMBAR to 4000h defines the supervisor access region to be 16 KB. As a result, for both the program memory map and the data memory map, the supervisor access region occupies the bottom half of the RAM space and the user access region occupies the upper 16 KB of the space.

- Program (PDB bus) accesses to RAM use the program memory map. Address accesses in the range of 06_0000h to 06_1FFFh in terms of words are supervisor mode accesses to the RAM. Accesses in the range of 06_2000h to 06_3FFFh in terms of words are user mode accesses to the RAM.
- Data (XAB1 or XAB2 bus) accesses to RAM use the data memory map. Address accesses in the range of 00_0000h to 00_1FFFh in terms of words are supervisor mode accesses to the RAM. Accesses in the range of 00_2000h to 00_3FFFh in terms of words are user mode accesses to the RAM.

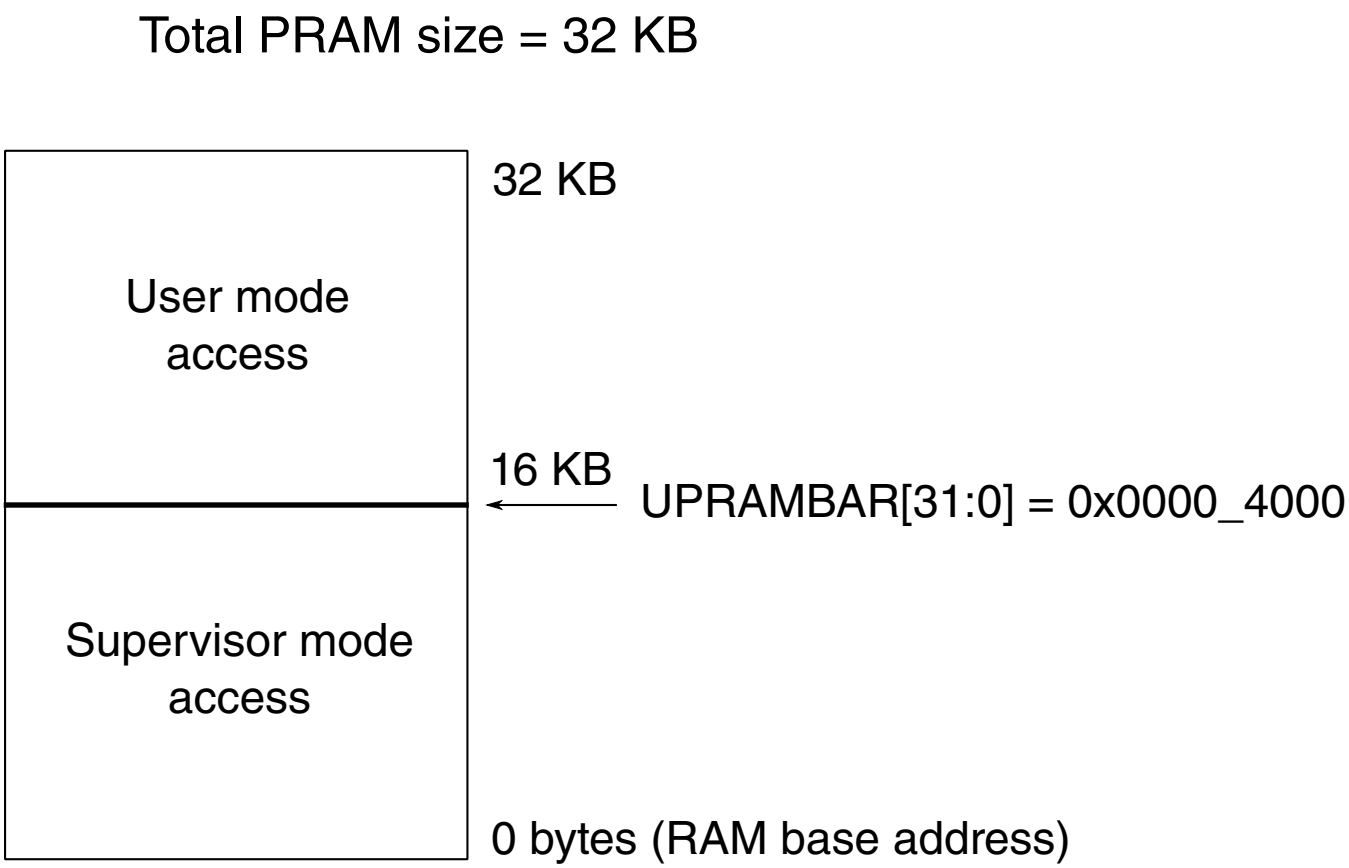


Figure 9-3. RAM Base Address Setup Example

9.7 Programming Example

To set up MRP and enter user mode, follow these required steps:

1. Initialize the UFLASHBAR, setting the base address of the user region in flash memory.
2. Initialize the UPRAMBAR, setting the base address of the user region in RAM.
3. Set up the user stack pointer via the MCM_SRPOSP register.
4. Enable resource protection using the MCM_RPCR[RPE] bit.
5. Push the desired user Status Register (SR) value and user Program Counter (PC) on the supervisor stack.
6. Execute a return from interrupt instruction (RTI, RTID, or FRTID), which reads the user Status Register (SR) value and user Program Counter (PC) from the supervisor stack to switch from supervisor to user mode.

It is recommended that user control is relinquished by the execution of an SWI instruction, returning control to the supervisor.

Chapter 10
Miscellaneous Control Module (MCM)

10.1 Introduction

The Miscellaneous Control Module (MCM) provides a myriad of miscellaneous control functions.

10.1.1 Features

The MCM provides the following:

- Program-visible information about the configuration and revision of the core and select system peripherals
- Registers for capturing information about core and core-peripheral bus errors, if enabled
- Control and configuration of memory resource protection (MRP)

10.2 Memory Map/Register Descriptions

Restriction

The MCM is a supervisor-only space. All MCM registers must be accessed by supervisor code.

In addition, each MCM register must be written in an access size equal to the register's width. For example, a 32-bit register must be written using a 32-bit access.

NOTE

The base address and offsets for these registers are presented in terms of bytes.

MCM memory map

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
1_8008	Crossbar switch (AXBS) slave configuration (MCM_PLASC)	16	R	000Fh	10.2.1/147
1_800A	Crossbar switch (AXBS) master configuration (MCM_PLAMC)	16	R	000Fh	10.2.2/147
1_800C	Core control register (MCM_CPCR)	32	R/W	0000_0000h	10.2.3/148
1_8010	Core fault address register (MCM_CFADR)	32	R	Undefined	10.2.4/149
1_8014	Core fault attributes register (MCM_CFATR)	8	R	Undefined	10.2.5/150
1_8015	Core fault location register (MCM_CFLOC)	8	R	00h	10.2.6/151
1_8016	Core fault interrupt enable register (MCM_CFIER)	8	R/W	00h	10.2.7/152
1_8017	MCM interrupt status register (MCM_CFISR)	8	R/W	00h	10.2.8/152
1_8018	Core fault data register (MCM_CFDTR)	32	R	Undefined	10.2.9/153
1_8020	Resource Protection Control Register (MCM_RPCR)	32	R/W	0000_0000h	10.2.10/153
1_8024	User Flash Base Address Register (MCM_UFLASHBAR)	32	R/W	0000_0000h	10.2.11/155
1_8028	User Program RAM Base Address Register (MCM_UPRAMBAR)	32	R/W	0000_0000h	10.2.12/155
1_8030	Resource Protection Other Stack Pointer (MCM_SRPOSP)	32	R/W	0000_0000h	10.2.13/156
1_8034	Memory Protection Illegal PC (MCM_SRPIPC)	32	R/W	0000_0000h	10.2.14/156
1_8038	Resource Protection Misaligned PC (MCM_SRPMPC)	32	R/W	0000_0000h	10.2.15/158

10.2.1 Crossbar switch (AXBS) slave configuration (MCM_PLASC)

The PLASC is a 16-bit read-only register identifying the presence/absence of bus slave connections to the device's Crossbar Switch (AXBS), plus a 1-bit flag defining the internal datapath width (DP64). The state of this register is defined by a module input signal; it can only be read from the programming model. Any attempted write is ignored.

Address: 1_8000h base + 8h offset = 1_8008h

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	DP64	0							ASC							
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1

MCM_PLASC field descriptions

Field	Description
15 DP64	Indicates if the datapath is 32 or 64 bits wide 0 Datapath width is 32 bits 1 Datapath width is 64 bits
14–8 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
ASC	Each bit in the ASC field indicates if there is a corresponding connection to the AXBS slave input port. For this device, this field always read 0x0F. 0 A bus slave connection to AXBS input port *n* is absent 1 A bus slave connection to AXBS input port *n* is present

10.2.2 Crossbar switch (AXBS) master configuration (MCM_PLAMC)

The PLAMC is a 16-bit read-only register identifying the presence/absence of bus master connections to the device's Crossbar Switch (AXBS). The state of this register is defined by a module input signal; it can only be read from the programming model. Any attempted write is ignored.

Address: 1_8000h base + Ah offset = 1_800Ah

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	0								AMC							
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1

MCM_PLAMC field descriptions

Field	Description
15–8 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
AMC	Each bit in the AMC field indicates if there is a corresponding connection to the AXBS master input port. For this device, this field always reads 0x0F. 0 A bus master connection to AXBS input port *n* is absent 1 A bus master connection to AXBS input port *n* is present

10.2.3 Core control register (MCM_CPCR)

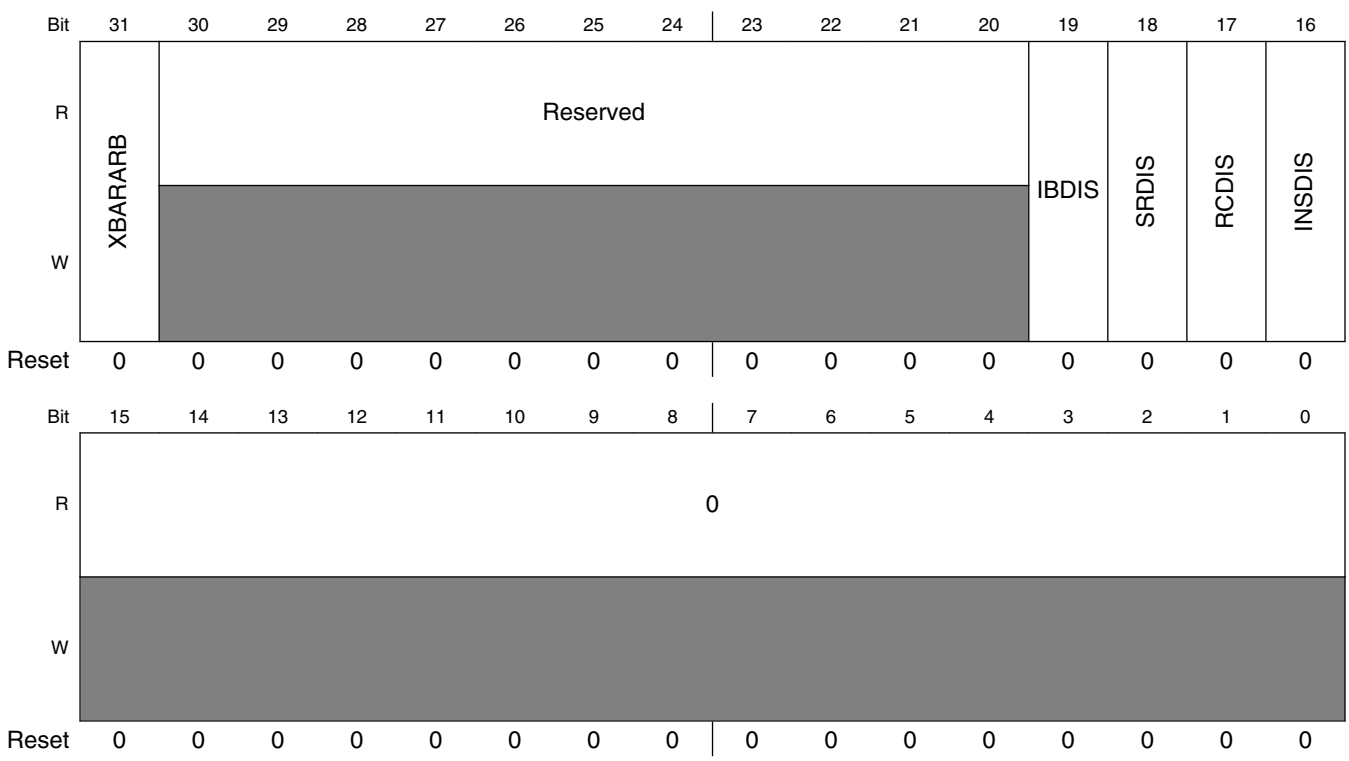
The 32-bit CPCR provides a program-visible register for user-defined control functions. Typically it controls the configuration of various chip-level modules. The upper word of this register is used to control core functions.

Restriction

This register must be written in a 32-bit access to change the core configuration.

The Test Bitfield instructions cannot be used on this register.

Address: 1_8000h base + Ch offset = 1_800Ch



MCM_CPCR field descriptions

Field	Description
31 XBARARB	Select DMA Controller priority in AXBS Crossbar Switch arbitration scheme Set the priority of the DMA Controller in the AXBS Crossbar Switch arbitration scheme. This device has 2 bus masters connected to the AXBS Crossbar Switch: the DMA Controller and the DSC core. For more information about AXBS Crossbar Switch arbitration, see Arbitration. 0 Fixed-priority arbitration is selected: DSC core has a higher priority than the DMA Controller's priority 1 Round-robin priority arbitration is selected: DMA Controller and DSC core have equal priority
30–20 Reserved	This read-only bitfield is reserved and is reset to zero. Do not write to this bitfield (write only zeros) or indeterminate results will occur. This field is reserved.
19 IBDIS	Disable core instruction buffer 0 Core longword instruction buffer enabled 1 Core longword instruction buffer disabled
18 SRDIS	Disable core new shadow region When this bit is 1, only the AGU shadow registers supported by the DSP56800E core are enabled. When this bit is 0, the additional AGU shadow registers on the DSP56800EX core are also enabled. 0 Core new shadow region enabled 1 Core new shadow region disabled
17 RCDIS	Disable core reverse carry When this bit is 0, the core supports bit-reverse addressing mode. When this bit is 1, the core does not support this mode. 0 Core reverse carry enabled 1 Core reverse carry disabled
16 INSDIS	Disable instructions supported only by DSP56800EX core The instructions supported only by the DSP56800EX core are the BFSC and 32-bit multiply and MAC instructions. 0 BFSC and 32-bit multiply and MAC instructions enabled 1 BFSC and 32-bit multiply and MAC instructions disabled
Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

10.2.4 Core fault address register (MCM_CFADR)

The CFADR is a read-only register indicating the address of the last core access terminated with an error response.

NOTE

This register is not initialized at reset, so its reset value is unknown.

Address: 1_8000h base + 10h offset = 1_8010h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	ADDR																															
W																																
Reset	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*

* Notes:
- x = Undefined at reset.

MCM_CFADR field descriptions

Field	Description
ADDR	Indicates the faulting address of the last core access terminated with an error response.

10.2.5 Core fault attributes register (MCM_CFATR)

The read-only CFATR register captures the processor's attributes of the last faulted core access to the system bus.

NOTE

This register is not initialized at reset, so its reset value is unknown.

Address: 1_8000h base + 14h offset = 1_8014h

Bit	7	6	5	4	3	2	1	0
Read	DIR	SIZE			0	BUFFER	1	TYPE
Write								
Reset	x*	x*	x*	x*	x*	x*	x*	x*

* Notes:
- x = Undefined at reset.

MCM_CFATR field descriptions

Field	Description
7 DIR	Direction of last faulted core access 0 Core read access 1 Core write access
6–4 SIZE	Size of last faulted core access 000 8-bit 001 16-bit 010 32-bit Else Reserved

Table continues on the next page...

MCM_CFATR field descriptions (continued)

Field	Description
3 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
2 BUFFER	Indicates if last faulted core access was bufferable 0 Non-bufferable 1 Bufferable
1 Reserved	This field is reserved. This read-only field is reserved and always has the value 1.
0 TYPE	Type of last faulted core access 0 Instruction 1 Data

10.2.6 Core fault location register (MCM_CFLOC)

The read-only CFLOC register indicates the location of the last captured fault.

Address: 1_8000h base + 15h offset = 1_8015h

Bit	7	6	5	4	3	2	1	0
Read	LOC		0					
Write								
Reset	0	0	0	0	0	0	0	0

MCM_CFLOC field descriptions

Field	Description
7–6 LOC	Location of last captured fault 00 Error occurred on M0 (instruction bus) 01 Error occured on M1 (operand A bus) 10 Error occured on M2 (operand B bus) 11 Reserved
Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

10.2.7 Core fault interrupt enable register (MCM_CFIER)

The CFIER register enables the system bus-error interrupt.

Address: 1_8000h base + 16h offset = 1_8016h

Bit	7	6	5	4	3	2	1	0
Read	ECFEI	0						
Write								
Reset	0	0	0	0	0	0	0	0

MCM_CFIER field descriptions

Field	Description
7 ECFEI	Enable core fault error interrupt 0 Do not generate an error interrupt on a faulted system bus cycle 1 Generate an error interrupt to the interrupt controller on a faulted system bus cycle
Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

10.2.8 MCM interrupt status register (MCM_CFISR)

This register indicates if a core fault interrupt has occured.

Address: 1_8000h base + 17h offset = 1_8017h

Bit	7	6	5	4	3	2	1	0
Read	CFEI	0						
Write	w1c							
Reset	0	0	0	0	0	0	0	0

MCM_CFISR field descriptions

Field	Description
7 CFEI	Core fault error interrupt flag Indicates if a bus fault has occurred. Writing a 1 clears this bit and negates the interrupt request. Writing a 0 has no effect. **NOTE:** This bit reports core faults regardless of the setting of CFIER[ECFEI]. Therefore, if the error interrupt is disabled and a core fault occurs, this bit is set. Then, if the error interrupt is subsequently enabled, an interrupt is immediately requested. To prevent an undesired interrupt, clear the captured error by writing one to CFEI before enabling the interrupt.

Table continues on the next page...

MCM_CFISR field descriptions (continued)

Field	Description
	0 No bus error 1 A bus error has occurred. The faulting address, attributes (and possibly write data) are captured in the CFADR, CFATR, and CFDTR registers. The error interrupt is enabled only if CFIER[ECFEI] is set.
Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

10.2.9 Core fault data register (MCM_CFDTR)

The CFDTR is a read-only register for capturing the data associated with the last faulted processor write data access from the device's internal bus. The CFDTR is valid only for faulted internal bus-write accesses; CFLOC[LOC] is cleared.

NOTE

This register is not initialized at reset, so its reset value is unknown.

Address: 1_8000h base + 18h offset = 1_8018h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	DATA																															
W																																
Reset	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*

* Notes:
- x = Undefined at reset.

MCM_CFDTR field descriptions

Field	Description
DATA	Contains write data associated with the faulting access of the last internal bus write access. The data value is taken directly from the write data bus. **NOTE** Read data is not captured.

10.2.10 Resource Protection Control Register (MCM_RPCR)

This register enables/disables memory resource protection (MRP) and locks/unlocks the values of the RP-related registers.

The core provides resource protection functionality. Refer to the detailed MRP description for more information about how to use the RPCR and other RP registers.

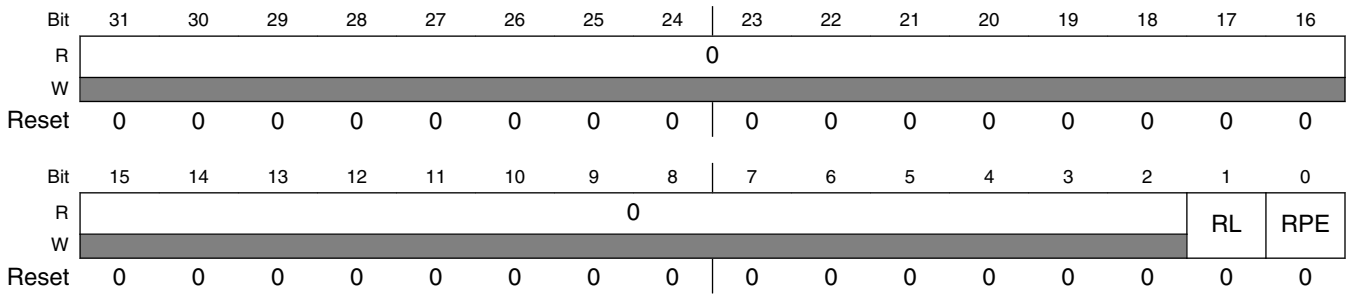
NOTE

The following write accesses to the resource protection registers are ignored and result in a bus error:

- Any non-32-bit write
- Any attempted write when resource protection hardware features are not enabled
- Any attempted write when RPCR[RL] is set (reads are allowed when RPCR[RL] is set)

The bus error interrupt must be enabled; otherwise, the errors are ignored by the core.

Address: 1_8000h base + 20h offset = 1_8020h



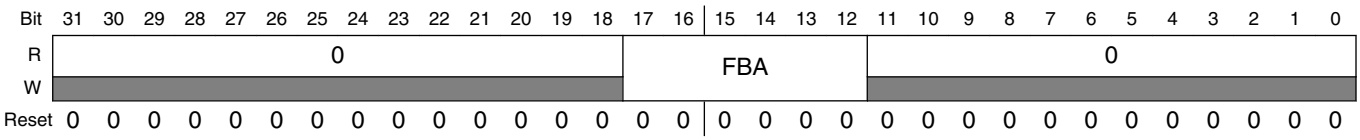
MCM_RPCR field descriptions

Field	Description
31–2 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1 RL	Register Lock This bit controls whether the values of the UFLASHBAR, UPRAMBAR, SRPOSP, SRPIPC, and SRPMPC registers can be modified. 0 RP register values may be changed 1 RP registers are locked and may not be changed until after a system reset
0 RPE	Resource Protection Enable 0 Resource protection disabled 1 Resource protection enabled

10.2.11 User Flash Base Address Register (MCM_UFLASHBAR)

This register defines the size of the portion of flash memory that is used for supervisor space when resource protection is enabled. The register can be used only when the RPCR[RPE] bit is 1, and its value can be changed only when the RPCR[RL] bit is 0.

Address: 1_8000h base + 24h offset = 1_8024h



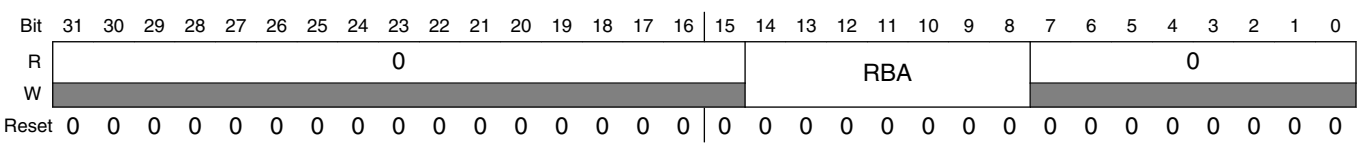
MCM_UFLASHBAR field descriptions

Field	Description
31–18 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
17–12 FBA	Flash Base Address for User Region Supports 4 KB granularity
Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

10.2.12 User Program RAM Base Address Register (MCM_UPRAMBAR)

This register defines the size of the portion of program RAM that is used for supervisor space when resource protection is enabled. The register can be used only when the RPCR[RPE] bit is 1, and its value can be changed only when the RPCR[RL] bit is 0.

Address: 1_8000h base + 28h offset = 1_8028h



MCM_UPRAMBAR field descriptions

Field	Description
31–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
14–8 RBA	Program RAM Base Address for User Region Supports 256 byte granularity

Table continues on the next page...

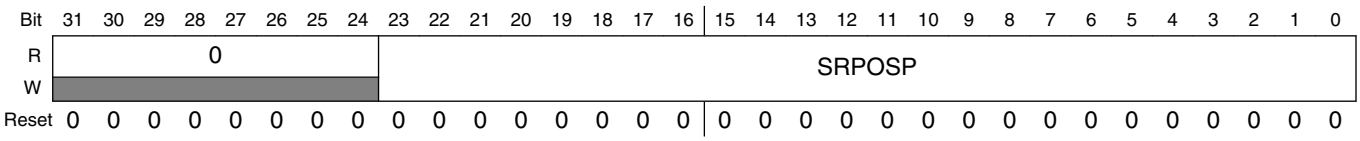
MCM_UPRAMBAR field descriptions (continued)

Field	Description
Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

10.2.13 Resource Protection Other Stack Pointer (MCM_SRPOSP)

This register can be used only when the RPCR[RPE] bit is 1, and its value can be changed only when the RPCR[RL] bit is 0.

Address: 1_8000h base + 30h offset = 1_8030h



MCM_SRPOSP field descriptions

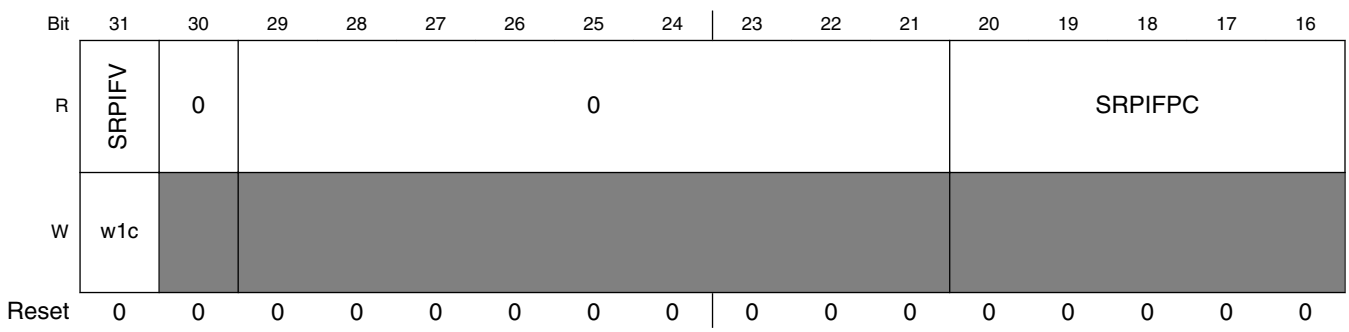
Field	Description
31–24 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
SRPOSP	Resource protection "other" SP

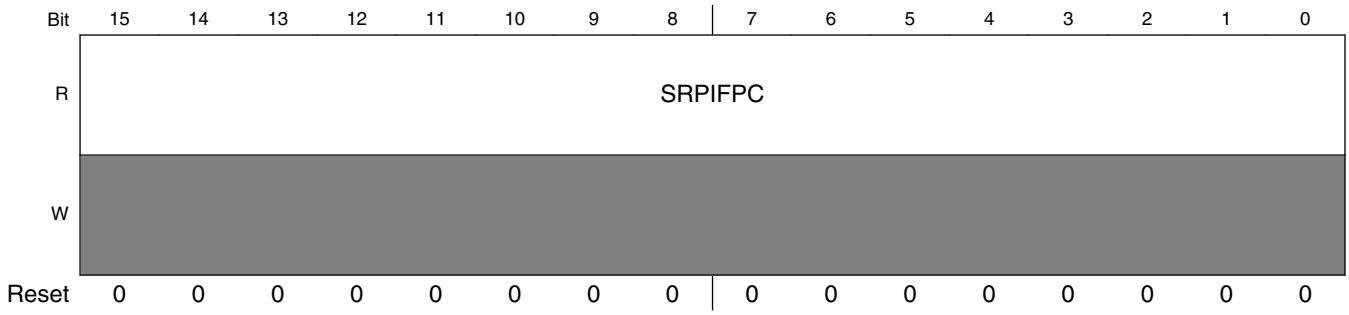
10.2.14 Memory Protection Illegal PC (MCM_SRPIPC)

This register can be used only when the RPCR[RPE] bit is 1, and its value can be changed only when the RPCR[RL] bit is 0.

This register's fault indicators apply only to faults resulting from supervisor and user access errors when resource protection is in effect. Other faults are not indicated in this register.

Address: 1_8000h base + 34h offset = 1_8034h





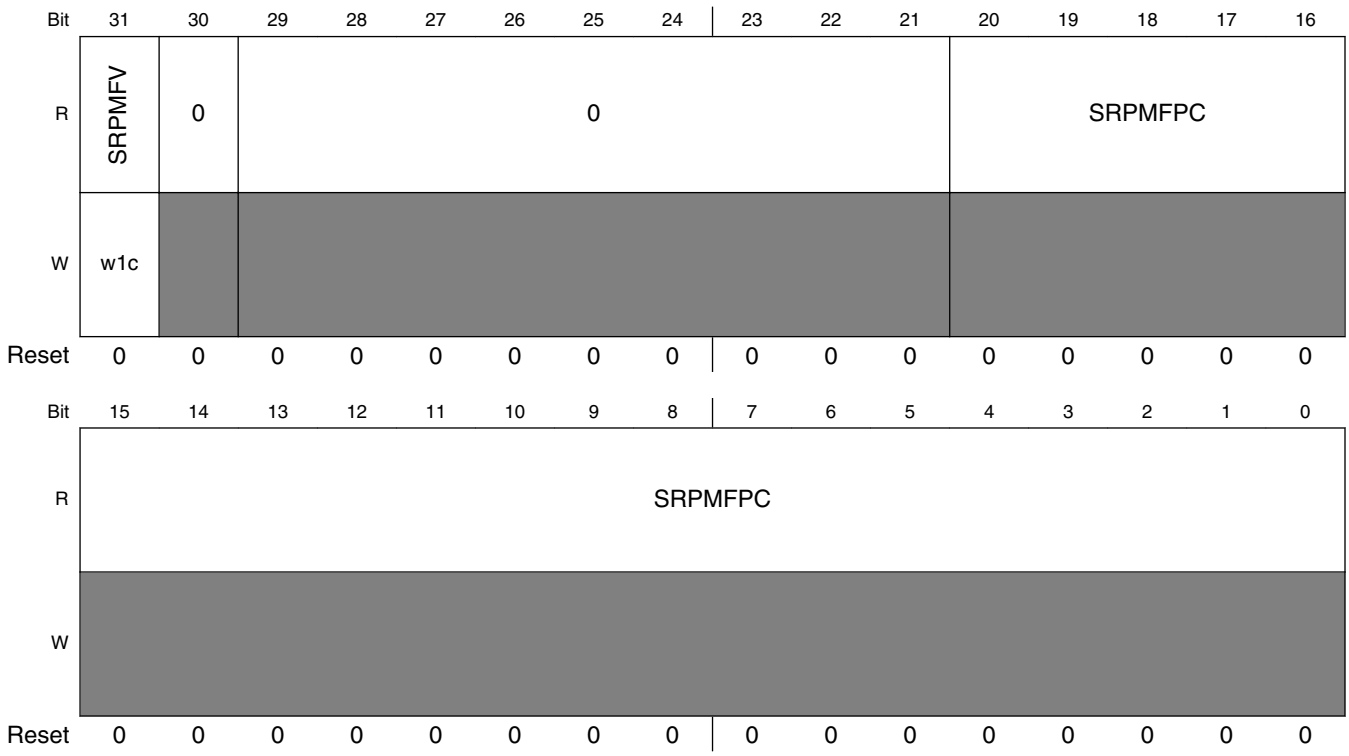
MCM_SRPIPC field descriptions

Field	Description
31 SRPIFV	Resource Protection Illegal Fault Valid When set, this bit indicates an RP illegal PC fault has occurred and the contents of SRPIFPC and SRPIFOR fields are valid. A write of 1 to this bit clears both the SRPIFOR and SRPIFV bits.
30 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
29–21 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
SRPIFPC	Resource Protection Illegal Faulting PC This is the 21-bit illegal faulting PC only for a resource protection fault.

10.2.15 Resource Protection Misaligned PC (MCM_SRPMPC)

This register's fault indicators apply only to faults resulting from supervisor and user access errors when resource protection is in effect. Other faults are not indicated in this register.

Address: 1_8000h base + 38h offset = 1_8038h



MCM_SRPMPC field descriptions

Field	Description
31 SRPMFV	Resource Protection Misaligned Fault Valid When set, this bit indicates an RP misaligned PC fault has occurred and the contents of the SRPMFPC and SRPMFOR fields are valid. A write of 1 to this bit clears both the SRPMFOR and SRPMFV bits.
30 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
29–21 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
SRPMFPC	Resource Protection Misaligned Faulting PC This is the 21-bit misaligned faulting PC only for a resource protection data access fault.

10.3 Functional Description

This section describes the functional description of MCM module.

10.3.1 Core Data Fault Recovery Registers

To aid in recovery from certain types of access errors, the MCM module supports a number of registers that capture access address, attribute, and data information on bus cycles terminated with an error response. These registers can then be read during the resulting exception service routine and the appropriate recovery performed.

The details on the core fault recovery registers are provided in the above sections. It is important to note these registers are used to capture fault recovery information on any processor-initiated system bus cycle terminated with an error.

Chapter 11
System Integration Module (SIM)

11.1 Introduction

This specification describes the operation and functionality of the System Integration Module for this device.

11.1.1 Overview

The System Integration Module (SIM) provides a variety of system control and status functions.

The system integration module is responsible for the following control/status functions:

- Reset control and status
- User accessible Software Control Registers which reset only at power on.
- Internal clock generation
- Implementation of STOP and WAIT low power modes and related clock gating
- System status registers
- Registers for software access to the JTAG ID of the chip and suggested trim values set at the factory
- Short addressing controls
- Test registers
- External and internal peripheral signal muxing control

11.1.2 Features

The SIM interacts with a variety of other on-chip resources and provides the following services:

- Controls and sequences the release of internal reset signals.
- Generates pipelined system clocks including management of holdoffs and core stalls.
- Generates peripheral clocks configurable over power modes.

- Manages low power mode entry and exit.
- Provides clock rate controls for selected peripherals.
- Selects the active peripheral function for IO when it is not used as GPIO.
- Provides write protection for safety critical memory mapped registers.
- Controls certain core functions, including the base for short addressing mode and enablement of the test access port (TAP) and debug mode entry as well as core low power modes.
- Controls voltage regulator.
- Selects CLKOUT clock source.
- Controls inter-peripheral muxing and signal relationships.

11.1.3 Modes of Operation

Since the SIM is responsible for distributing clocks and resets across the chip, it must understand the various chip operating modes and take appropriate action. These include:

- **RESET Modes:**This is the sequencing of reset deassertion as part comes out of reset.
  - **Clock Reset Mode**: The processor, all peripherals, and the CLKGEN module are all in reset.
  - **System and Core Reset Mode:** Clocks activate while the core and peripherals remain in reset.
  - **Core-Only Reset Mode:** core in reset while peripherals are activated.

NOTE

Core-only reset mode is required to provide time for the on-chip flash interface units to load part configuration data from flash and establish the boot address.

- **RUN Mode:** This is the primary mode of operation for this device. In this mode, the processor clocks, system clocks, and all enabled peripheral clocks are operational.
- **Debug Mode:** The processor is in debug mode (controlled via JTAG/EOnCE). All system and peripheral clocks with the exception of the COP and PWM's continue to run. The COP is disabled in debug mode and PWM outputs are optionally disabled (see the PWM details) to bypass undesired inverter operations.
- **WAIT Mode:** In WAIT mode, the core clock and system clocks are disabled but all enabled peripheral clocks continue to operate. The COP can optionally be stopped in WAIT mode. Similarly, the PWM outputs can optionally be switched off in WAIT mode to disable inverter from being driven.
- **STOP Mode:** In STOP mode, the core clock and system clocks are disabled. Individual peripheral clocks are also disabled unless specifically configured in the SIM to continue running in STOP mode (useful for generating STOP recovery

interrupts from selected devices). The COP can optionally be stopped in STOP mode. If desired, the OCCS may be configured to disable the PLL and/or select a lower frequency clock source prior to entering STOP mode.

NOTE

The power management controller (PMC) provides additional control of power consumption by supporting low-power states with reduced regulator drive capacity. The on-chip clock system (OCCS) module also influences power consumption by controlling the operating frequency of the system and peripheral clocks and by supporting the powering down of unused clock sources.

11.1.4 Block Diagram

The following diagram depicts the System Integration Module.

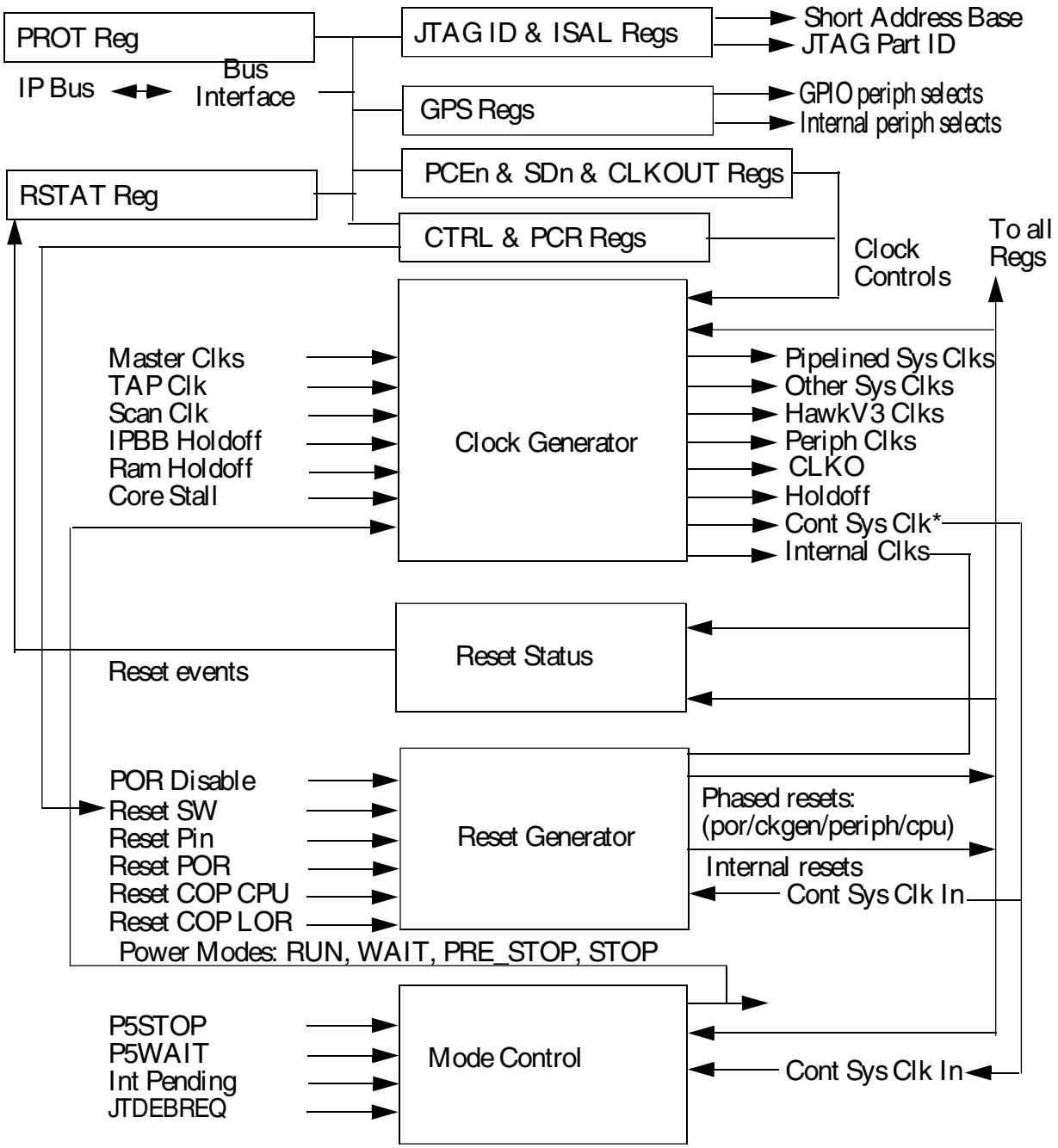


Figure 11-1. SIM Block Diagram

11.2 Memory Map and Register Descriptions

A write to an address without an associated register is a NOP. A read from an address without an associated register returns unknown data.

SIM memory map

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
E400	Control Register (SIM_CTRL)	16	R/W	00C0h	11.2.1/166
E401	Reset Status Register (SIM_RSTAT)	16	R	0004h	11.2.2/168
E402	Software Control Register (SIM_SCR0)	16	R/W	0000h	11.2.3/169
E403	Software Control Register (SIM_SCR1)	16	R/W	0000h	11.2.4/170
E404	Software Control Register (SIM_SCR2)	16	R/W	0000h	11.2.5/170
E405	Software Control Register (SIM_SCR3)	16	R/W	0000h	11.2.6/170
E406	Most Significant Half of JTAG ID (SIM_MSHID)	16	R	16B4h	11.2.7/171
E407	Least Significant Half of JTAG ID (SIM_LSHID)	16	R	801Dh	11.2.8/172
E408	Power Control Register (SIM_PWR)	16	R/W	0000h	11.2.9/173
E40A	Clock Output Select Register (SIM_CLKOUT)	16	R/W	1020h	11.2.10/174
E40B	Peripheral Clock Rate Register (SIM_PCR)	16	R/W	0000h	11.2.11/176
E40C	Peripheral Clock Enable Register 0 (SIM_PCE0)	16	R/W	0000h	11.2.12/177
E40D	Peripheral Clock Enable Register 1 (SIM_PCE1)	16	R/W	0000h	11.2.13/179
E40E	Peripheral Clock Enable Register 2 (SIM_PCE2)	16	R/W	0000h	11.2.14/181
E40F	Peripheral Clock Enable Register 3 (SIM_PCE3)	16	R/W	0000h	11.2.15/182
E410	STOP Disable Register 0 (SIM_SD0)	16	R/W	0000h	11.2.16/183
E411	Peripheral Clock STOP Disable Register 1 (SIM_SD1)	16	R/W	0000h	11.2.17/186
E412	Peripheral Clock STOP Disable Register 2 (SIM_SD2)	16	R/W	0000h	11.2.18/188
E413	Peripheral Clock STOP Disable Register 3 (SIM_SD3)	16	R/W	0000h	11.2.19/190
E414	I/O Short Address Location Register (SIM_IOSAHI)	16	R/W	0000h	11.2.20/192
E415	I/O Short Address Location Register (SIM_IOSALO)	16	R/W	0380h	11.2.21/193
E416	Protection Register (SIM_PROT)	16	R/W	0000h	11.2.22/194
E417	GPIOA LSBs Peripheral Select Register (SIM_GPSAL)	16	R/W	0000h	11.2.23/196
E419	GPIOC LSBs Peripheral Select Register (SIM_GPSCL)	16	R/W	0000h	11.2.24/197
E41A	GPIOC MSBs Peripheral Select Register (SIM_GPSCH)	16	R/W	0000h	11.2.25/198
E41C	GPIOE LSBs Peripheral Select Register (SIM_GPSEL)	16	R/W	0000h	11.2.26/200
E41E	GPIOF LSBs Peripheral Select Register (SIM_GPSFL)	16	R/W	0000h	11.2.27/201
E41F	GPIOF MSBs Peripheral Select Register (SIM_GPSFH)	16	R/W	0000h	11.2.28/202
E422	Internal Peripheral Select Register 0 (SIM_IPS0)	16	R/W	0000h	11.2.29/202

Table continues on the next page...

SIM memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
E423	Miscellaneous Register 0 (SIM_MISC0)	16	R/W	0000h	11.2.30/205
E424	Peripheral Software Reset Register 0 (SIM_PSWR0)	16	R/W	0000h	11.2.31/206
E425	Peripheral Software Reset Register 1 (SIM_PSWR1)	16	R/W	0000h	11.2.32/207
E426	Peripheral Software Reset Register 2 (SIM_PSWR2)	16	R/W	0000h	11.2.33/208
E427	Peripheral Software Reset Register 3 (SIM_PSWR3)	16	R/W	0000h	11.2.34/210
E428	Power Mode Register (SIM_PWRMODE)	16	R/W	0000h	11.2.35/211
E42C	Non-Volatile Memory Option Register 2 (High) (SIM_NVMOPT2H)	16	R	0220h	11.2.36/212
E42D	Non-Volatile Memory Option Register 2 (Low) (SIM_NVMOPT2L)	16	R	7100h	11.2.37/213
E42E	Non-Volatile Memory Option Register 3 (High) (SIM_NVMOPT3H)	16	R	0000h	11.2.38/213

11.2.1 Control Register (SIM_CTRL)

Address: E400h base + 0h offset = E400h

Bit	15	14	13	12	11	10	9	8
Read	0					RST_FILT	0	DMAEbl
Write								
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
Read Write	DMAEbl		OnceEbl	SWRst	STOP_disable		WAIT_disable	
Reset	1	1	0	0	0	0	0	0

SIM_CTRL field descriptions

Field	Description
15–11 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
10 RST_FILT	External Reset Padcell Input Filter Enable This input controls an optional analog input filter on the padcell supporting the external reset input function. When enabled, the filter removes transient signals on the input at the expense of an increased input delay. When enabled, the filter affects all input functions supported by that padcell, including GPIO. This bit is reset on POR only. The filter has two basic applications. When this pad is configured as an output function such as a GPIO output and the part is reset, the filter can remove transients that might result in a false indication of an external reset assertion in the SIM's RSTAT register. When this padcell is configured as the external reset input, the filter can filter out noise on the external reset to reduce the chance of an unintended external reset assertion. The filter delay should be considered before enabling the filter, especially for safety critical applications.

Table continues on the next page...

SIM_CTRL field descriptions (continued)

Field	Description
	0 Input filter on external reset disabled 1 Input filter on external reset enabled
9 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

8–6
DMAEbl
	DMA Enable
	
	This field controls whether DMA is enabled in RUN, WAIT, RUN and WAIT, or all modes. The DMA functions as a system bus master capable of performing IO on any address in data space. It also contains memory mapped registers through which it is configured. The DMA must be enabled to function either as a system bus master or to perform IO to its memory mapped registers.
	
	NOTE: The DMA incorporates synchronous reset logic and must therefore be powered on at reset. It may subsequently be powered down at any time when not in use.
	
	NOTE: Entry to stop mode affects in-progress DMA transactions differently than does entry to wait mode:
	• When the DMA module is configured to be disabled in stop mode: If any DMA transaction is in progress during stop mode entry, the transaction will complete before the MCU gates the DMA controller's clock.
	• When the DMA module is configured to be disabled in wait mode:
	  • If any DMA transaction is in progress during wait mode entry, the transaction might be incomplete and end prematurely when the MCU gates the DMA controller's clock.
	  • To avoid data loss, the user's application must ensure no DMA transaction is in progress during the entry to wait mode.
	
	000 DMA module is disabled.
	001 DMA module is enabled in run mode only.
	010 DMA module is enabled in run and wait modes only.
	011 DMA module is enabled in all power modes.
	100 DMA module is disabled and the DMAEbl field is write protected until the next reset.
	101 DMA module is enabled in run mode only and the DMAEbl field is write protected until the next reset.
	110 DMA module is enabled in run and wait modes only and the DMAEbl field is write protected until the next reset.
	111 DMA module is enabled in all low power modes and the DMAEbl field is write protected until the next reset.

5 OnceEbl	OnCE Enable 0 The OnCE clock to the core is enabled when the core TAP is enabled. 1 The OnCE clock to the core is always enabled.
4 SWRst	SOFTWARE RESET Writing a 1 to this field causes a device reset.
3–2 STOP_disable	STOP Disable 00 Stop mode is entered when the core executes a STOP instruction. 01 The core STOP instruction does not cause entry into stop mode. 10 Stop mode is entered when the core executes a STOP instruction, and the STOP_disable field is write protected until the next reset. 11 The core STOP instruction does not cause entry into stop mode, and the STOP_disable field is write protected until the next reset.
WAIT_disable	WAIT Disable

Table continues on the next page...

SIM_CTRL field descriptions (continued)

Field	Description
	00 Wait mode is entered when the core executes a WAIT instruction. 01 The core WAIT instruction does not cause entry into wait mode. 10 Wait mode is entered when the core executes a WAIT instruction, and the WAIT_disable field is write protected until the next reset. 11 The core WAIT instruction does not cause entry into wait mode, and the WAIT_disable field is write protected until the next reset.

11.2.2 Reset Status Register (SIM_RSTAT)

This register is updated upon any system reset and indicates the cause of the most recent reset. It also controls whether the COP reset vector or regular reset vector in the vector table is used. This register is asynchronously reset during power-on reset and subsequently is synchronously updated based on the level of the external reset, software reset, or COP reset inputs. It is "one-hot encoded," which means that only one reset source is indicated at any given time. When multiple reset sources assert simultaneously, the reset source with the highest precedence is indicated in this register. . The precedence from highest to lowest is POR, EXTR, COP_LOR, COP_CPU, and SWR. POR is always set during a power-on reset; however, POR is cleared and EXTR is set if the external reset pin is asserted or remains asserted after the power-on reset has deasserted.

Address: E400h base + 1h offset = E401h

Bit	15	14	13	12	11	10	9	8
Read	0							
Write								
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
Read	Reserved	SWR	COP_CPU	COP_LOR	EXTR	POR	0	
Write								
Reset	0	0	0	0	0	1	0	0

SIM_RSTAT field descriptions

Field	Description
15–8 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
7 Reserved	This field is reserved.

Table continues on the next page...

SIM_RSTAT field descriptions (continued)

Field	Description
6 SWR	Software Reset When set, this bit indicates that the previous system reset occurred as a result of a software reset (1 written to the SWRst bit of the SIM's CTRL register). SWR is not set if a COP reset, external reset, or POR also occurred.
5 COP_CPU	COP CPU Time-out Reset When set, this bit indicates that the previous system reset was caused when the computer operating properly (COP) module signaled a CPU time-out reset. COP_CPU is not set if an external reset, POR, or COP loss of reference reset also occurred. If COP_CPU is set as code starts executing, the COP reset vector in the vector table is used. Otherwise, the normal reset vector is used.
4 COP_LOR	COP Loss of Reference Reset When set, this bit indicates that the previous system reset was caused when the computer operating properly (COP) module signaled a loss of reference clock reset. COP_LOR is not set if an external or POR also occurred. If COP_LOR is set as code starts executing, the COP reset vector in the vector table is used. Otherwise, the normal reset vector is used.
3 EXTR	External Reset When set, this bit indicates that the previous system reset was caused by an external reset. EXTR is set only if the external reset pin was asserted or remained asserted after the power-on reset deasserted.
2 POR	Power-on Reset This bit is set by a power-on reset.
Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

11.2.3 Software Control Register (SIM_SCR0)

Address: E400h base + 2h offset = E402h

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read Write	SCR0															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SIM_SCR0 field descriptions

Field	Description
SCR0	Software Control Data This field is for general-purpose use by software. It is reset only by a power-on reset.

11.2.4 Software Control Register (SIM_SCR1)

Address: E400h base + 3h offset = E403h

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read Write	SCR1															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SIM_SCR1 field descriptions

Field	Description
SCR1	Software Control Data This field is for general-purpose use by software. It is reset only by a power-on reset.

11.2.5 Software Control Register (SIM_SCR2)

Address: E400h base + 4h offset = E404h

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read Write	SCR2															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SIM_SCR2 field descriptions

Field	Description
SCR2	Software Control Data This field is for general-purpose use by software. It is reset only by a power-on reset.

11.2.6 Software Control Register (SIM_SCR3)

Address: E400h base + 5h offset = E405h

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read Write	SCR3															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SIM_SCR3 field descriptions

Field	Description
SCR3	Software Control Data This field is for general-purpose use by software. It is reset only by a power-on reset.

11.2.7 Most Significant Half of JTAG ID (SIM_MSHID)

This read-only register returns the most significant half of the JTAG ID for the device.

Address: E400h base + 6h offset = E406h

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

SIM_MSHID field descriptions

Field	Description
15–13 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
12 Reserved	This field is reserved. This read-only field is reserved and always has the value 1.
11 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
10 Reserved	This field is reserved. This read-only field is reserved and always has the value 1.
9 Reserved	This field is reserved. This read-only field is reserved and always has the value 1.
8 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
7 Reserved	This field is reserved. This read-only field is reserved and always has the value 1.
6 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
5 Reserved	This field is reserved. This read-only field is reserved and always has the value 1.
4 Reserved	This field is reserved. This read-only field is reserved and always has the value 1.
3 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
2 Reserved	This field is reserved. This read-only field is reserved and always has the value 1.
Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

11.2.8 Least Significant Half of JTAG ID (SIM_LSHID)

This read-only register returns the least significant half of the JTAG ID for the device.

Address: E400h base + 7h offset = E407h

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

SIM_LSHID field descriptions

Field	Description
15 Reserved	This field is reserved. This read-only field is reserved and always has the value 1.
14–5 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
4 Reserved	This field is reserved. This read-only field is reserved and always has the value 1.
3 Reserved	This field is reserved. This read-only field is reserved and always has the value 1.
2 Reserved	This field is reserved. This read-only field is reserved and always has the value 1.
1 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 1.

11.2.9 Power Control Register (SIM_PWR)

This register contains control fields used to enable/disable the standby and powerdown modes of the large and small voltage regulators in the Power Management Controller module. The large regulator supplies digital standard cell core logic. The small regulator 2.7 V supply powers digital logic in hard blocks. The register independently controls the regulator mode. However, if the FTFL module's FOPT[0] bit is 1 (reflecting a power mode selection via the Flash Option register), writing to these control bits does not affect the regulators.

Address: E400h base + 8h offset = E408h

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	0								SR12STDBY		SR27PDN		SR27STDBY		LRSTDBY	
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SIM_PWR field descriptions

Field	Description
15–8 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
7–6 SR12STDBY	Small Regulator 1.2 V Supply Standby Control This field controls the standby mode of the 1.2 V supply from the small voltage regulator. Standby mode has restricted drive capacity but substantially reduces power consumption. The field value can be optionally write protected. 00 Small regulator 1.2 V supply placed in normal mode (default). 01 Small regulator 1.2 V supply placed in standby mode. 10 Small regulator 1.2 V supply placed in normal mode and SR12STDBY is write protected until chip reset. 11 Small regulator 1.2 V supply placed in standby mode and SR12STDBY is write protected until chip reset.
5–4 SR27PDN	Small Regulator 2.7 V Supply Powerdown Control This field controls the powerdown mode of the 2.7 V supply from the small voltage regulator. Powerdown mode shuts down the 2.7 V regulated supply from the small regulator and eliminates its power consumption. Analog modules powered by this supply should themselves be powered down before entering this mode. These controls are located in the OCCS module.. 00 Small regulator placed in normal mode (default). 01 Small regulator placed in powerdown mode. 10 Small regulator placed in normal mode and SR27PDN is write protected until chip reset. 11 Small regulator placed in powerdown mode and SR27PDN is write protected until chip reset.
3–2 SR27STDBY	Small Regulator 2.7 V Supply Standby Control

Table continues on the next page...

SIM_PWR field descriptions (continued)

Field	Description
	This field controls the standby mode of the 2.7 V supply from the small voltage regulator. Standby mode has restricted drive capacity but substantially reduces power consumption. The field value can be optionally write protected. 00 Small regulator 2.7 V supply placed in normal mode (default). 01 Small regulator 2.7 V supply placed in standby mode. 10 Small regulator 2.7 V supply placed in normal mode and SR27STDBY is write protected until chip reset. 11 Small regulator 2.7 V supply placed in standby mode and SR27STDBY is write protected until chip reset.
LRSTDBY	Large Regulator Standby Control This field controls the standby mode of the primary on-device voltage regulator. Standby mode reduces overall device power consumption but places significant constraints on the allowed operating frequency. Refer to the Power Management Controller module's description for details on standby mode. The field value can optionally be write protected. 00 Large regulator placed in normal mode (default). 01 Large regulator placed in standby mode. 10 Large regulator placed in normal mode and LRSTDBY is write protected until device reset. 11 Large regulator placed in standby mode and LRSTDBY is write protected until device reset.

11.2.10 Clock Output Select Register (SIM_CLKOUT)

This register can be used to multiplex selected clocks generated inside the clock generation, SIM, and other internal modules onto the SIM CLKOUT clock output signal. This signal, in turn, is typically available to be brought out to an external pad. Glitches may be produced when the clock is enabled or switched. The delay from the clock source to the output is unspecified. The visibility of the waveform on the CLKOUT on an external pad is subject to the frequency limitations of the associated I/O cell.

Address: E400h base + Ah offset = E40Ah

Bit	15	14	13	12	11	10	9	8
Read Write	CLKODIV			CLKDIS1	Reserved		CLKOSEL1	
Reset	0	0	0	1	0	0	0	0

Bit	7	6	5	4	3	2	1	0
Read Write	CLKOSEL1	Reserved	CLKDIS0	Reserved			CLKOSEL0	
Reset	0	0	1	0	0	0	0	0

SIM_CLKOUT field descriptions

Field	Description
15–13 CLKODIV	CLKOUT divide factor Configures dividers on the CLKOUT0 and CLKOUT1 outputs used to reduce output frequencies to levels supported by their respective pad cells. 000 Divide by 1 001 Divide by 2 010 Divide by 4 011 Divide by 8 100 Divide by 16 101 Divide by 32 110 Divide by 64 111 Divide by 128
12 CLKDIS1	Disable for CLKOUT1 0 CLKOUT1 output is enabled and outputs the signal indicated by CLKOSEL1 1 CLKOUT1 is disabled
11–10 Reserved	This field is reserved. Always write 0 to this field for normal operation.
9–7 CLKOSEL1	CLKOUT1 Select Selects the clock source for the CLKOUT1 pin. The internal delay to the CLKOUT1 output is unspecified. The signal at the output pad is undefined when the CLKOUT1 signal frequency exceeds the rated frequency of the IO cell. CLKOUT1 may glitch when CLKDIS1, CLKOSEL1, or CLKODIV settings are changed. **NOTE:** Propagation of CLKOUT1 to an external pad requires the proper setting of the related GPSn and GPIOn_PER fields. The following settings define the signal to be output on CLKOUT1 based on the setting of the associated CLKDIS and CLKOSEL fields. 000 Function = BUS_CLK continuous after reset 001 Function = 2X_BUS_CLK continuous after reset 010 Function = DIV4_BUS_CLK continuous after reset 011 Function = MSTR_OSC (master clock) continuous after reset 100 Function = ROSC_8M (8 MHz / 400 kHz relaxation oscillator clock) 101 Function = ROSC_32K (32 kHz relaxation oscillator clock) 110 Reserved. For normal operation, do not write 11x. 111 Reserved. For normal operation, do not write 11x.
6 Reserved	This field is reserved. Always write 0 to this bit for normal operation.
5 CLKDIS0	Disable for CLKOUT0 0 CLKOUT0 output is enabled and outputs the signal indicated by CLKOSEL0 1 CLKOUT0 is disabled
4–3 Reserved	This field is reserved. Always write 0 to this field for normal operation.
CLKOSEL0	CLKOUT0 Select

Table continues on the next page...

SIM_CLKOUT field descriptions (continued)

Field	Description
	Selects the clock source for the CLKOUT0 pin. The internal delay to the CLKOUT0 output is unspecified. The signal at the output pad is undefined when the CLKOUT0 signal frequency exceeds the rated frequency of the IO cell. CLKOUT0 may glitch when CLKDIS0, CLKOSEL0, or CLKODIV settings are changed. **NOTE:** Propagation of CLKOUT0 to an external pad requires the proper setting of the related GPSn and GPIOn_PER fields. The following settings define the signal to be output on CLKOUT0 based on the setting of the associated CLKDIS and CLKOSEL fields. 000 Function = BUS_CLK continuous after reset 001 Function = 2X_BUS_CLK continuous after reset 010 Function = DIV4_BUS_CLK continuous after reset 011 Function = MSTR_OSC (master clock) continuous after reset 100 Function = ROSC_8M (8 MHz / 400 kHz relaxation oscillator clock) 101 Function = ROSC_32K (32 kHz relaxation oscillator clock) 110 Reserved. For normal operation, do not write 11x. 111 Reserved. For normal operation, do not write 11x.

11.2.11 Peripheral Clock Rate Register (SIM_PCR)

By default, all peripherals are clocked at the system clock rate. Selected peripherals clocks can be clocked at two times this normal rate. This register enables high-speed clocking for peripherals that support it. High-speed peripheral clocking is dependent on the mstr_2x clock output of the OCCS.

Peripherals should not be left in an enabled or operating mode while reconfiguring their clocks using the controls in the SIM or OCCS module. PCR bits should therefore be changed only while the applicable peripheral is disabled. Refer to the peripheral's description for details.

When a peripheral operates in high-speed mode, the I/O rate to the peripheral remains limited to the system clock rate because that is the rate at which the processor operates. For peripherals with a single clock input, that clock operates at the high-speed rate and a high-speed I/O gasket is used to coordinate I/O with the processor. For peripherals with separate I/O and "run" clocks, the I/O clock operates at the normal peripheral clock rate and only the "run" clock operates at the 2x high-speed rate.

Address: E400h base + Bh offset = E40Bh

Bit	15	14	13	12	11	10	9	8
Read	0		SCI0_CR	SCI1_CR	0	0		
Write								
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
Read	0							
Write								
Reset	0	0	0	0	0	0	0	0

SIM_PCR field descriptions

Field	Description
15–14 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
13 SCI0_CR	SCI0 Clock Rate This bit selects the clock speed for the SCI0 module. 0 SCI0 clock rate equals core clock rate (default) 1 SCI0 clock rate equals 2X core clock rate
12 SCI1_CR	SCI1 Clock Rate This bit selects the clock speed for the SCI1 module. 0 SCI1 clock rate equals core clock rate (default) 1 SCI1 clock rate equals 2X core clock rate
11 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

11.2.12 Peripheral Clock Enable Register 0 (SIM_PCE0)

The Peripheral Clock Enable registers enable or disable clocking of individual peripherals. Enabling the clocks of only peripherals that are in use can achieve significant power savings. When a peripheral's clock is disabled, no functionality, including I/O, is available to that peripheral.

Peripherals should not be left in an enabled or operating mode while their clocks are disabled or while their clocks are reconfigured using the controls in the SIM or OCCS module. PCE bits should therefore be changed only while the applicable peripheral is disabled. Refer to the peripheral's description for further details.

Enabled peripheral clocks still become disabled in STOP mode unless the peripheral's STOP Disable control in the SDn registers is set to 1.

NOTE

The FTFL (flash) module does not have a PCE bit because it does not have a separate, distinct peripheral clock.

Address: E400h base + Ch offset = E40Ch

Bit	15	14	13	12	11	10	9	8
Read Write	TA0	TA1	TA2	TA3	TB0	TB1	TB2	TB3
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
Read	0	GPIOA	GPIOB	GPIOC	GPIOD	GPIOE	GPIOF	0
Write								
Reset	0	0	0	0	0	0	0	0

SIM_PCE0 field descriptions

Field	Description
15 TA0	TMRA0 IPBus Clock Enable 0 The peripheral is not clocked. 1 The peripheral is clocked.
14 TA1	TMRA1 IPBus Clock Enable 0 The peripheral is not clocked. 1 The peripheral is clocked.
13 TA2	TMRA2 IPBus Clock Enable 0 The peripheral is not clocked. 1 The peripheral is clocked.
12 TA3	TMRA3 IPBus Clock Enable 0 The peripheral is not clocked. 1 The peripheral is clocked.
11 TB0	TMRB0 IPBus Clock Enable 0 The peripheral is not clocked. 1 The peripheral is clocked.
10 TB1	TMRB1 IPBus Clock Enable 0 The peripheral is not clocked. 1 The peripheral is clocked.
9 TB2	TMRB2 IPBus Clock Enable 0 The peripheral is not clocked. 1 The peripheral is clocked.
8 TB3	TMRB3 IPBus Clock Enable 0 The peripheral is not clocked. 1 The peripheral is clocked.
7 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
6 GPIOA	GPIOA IPBus Clock Enable 0 The peripheral is not clocked. 1 The peripheral is clocked.

Table continues on the next page...

SIM_PCE0 field descriptions (continued)

Field	Description
5 GPIOB	GPIOB IPBus Clock Enable 0 The peripheral is not clocked. 1 The peripheral is clocked.
4 GPIOC	GPIOC IPBus Clock Enable 0 The peripheral is not clocked. 1 The peripheral is clocked.
3 GPIOD	GPIOD IPBus Clock Enable 0 The peripheral is not clocked. 1 The peripheral is clocked.
2 GPIOE	GPIOE IPBus Clock Enable 0 The peripheral is not clocked. 1 The peripheral is clocked.
1 GPIOF	GPIOF IPBus Clock Enable 0 The peripheral is not clocked. 1 The peripheral is clocked.
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

11.2.13 Peripheral Clock Enable Register 1 (SIM_PCE1)

The Peripheral Clock Enable registers enable or disable clocking of individual peripherals. Enabling the clocks of only peripherals that are in use can achieve significant power savings. When a peripheral's clock is disabled, no functionality, including I/O, is available to that peripheral.

Peripherals should not be left in an enabled or operating mode while their clocks are disabled or while their clocks are reconfigured using the controls in the SIM or OCCS module. PCE bits should therefore be changed only while the applicable peripheral is disabled. Refer to the peripheral's description for further details.

Enabled peripheral clocks still become disabled in STOP mode unless the peripheral's STOP Disable control in the SDn registers is set to 1.

Address: E400h base + Dh offset = E40Dh

Bit	15	14	13	12	11	10	9	8
Read	0		DAC	SCI0	SCI1	0	QSPI0	0
Write								
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
Read	0	IIC0	IIC1	0				FLEXCAN
Write								
Reset	0	0	0	0	0	0	0	0

SIM_PCE1 field descriptions

Field	Description
15–14 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
13 DAC	DAC IPBus Clock Enable 0 The peripheral is not clocked. 1 The peripheral is clocked.
12 SCI0	SCI0 IPBus Clock Enable 0 The peripheral is not clocked. 1 The peripheral is clocked.
11 SCI1	SCI1 IPBus Clock Enable 0 The peripheral is not clocked. 1 The peripheral is clocked.
10 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
9 QSPI0	QSPI0 IPBus Clock Enable 0 The peripheral is not clocked. 1 The peripheral is clocked.
8 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
7 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
6 IIC0	IIC0 IPBus Clock Enable 0 The peripheral is not clocked. 1 The peripheral is clocked.
5 IIC1	IIC1 IPBus Clock Enable 0 The peripheral is not clocked. 1 The peripheral is clocked.
4–1 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
0 FLEXCAN	FlexCAN IPBus Clock Enable 0 The peripheral is not clocked. 1 The peripheral is clocked.

11.2.14 Peripheral Clock Enable Register 2 (SIM_PCE2)

The Peripheral Clock Enable registers enable or disable clocking of individual peripherals. Enabling the clocks of only peripherals that are in use can achieve significant power savings. When a peripheral's clock is disabled, no functionality, including I/O, is available to that peripheral.

Peripherals should not be left in an enabled or operating mode while their clocks are disabled or while their clocks are reconfigured using the controls in the SIM or OCCS module. PCE bits should therefore be changed only while the applicable peripheral is disabled. Refer to the peripheral's description for further details.

Enabled peripheral clocks still become disabled in STOP mode unless the peripheral's STOP Disable control in the SDn registers is set to 1.

Address: E400h base + Eh offset = E40Eh

Bit	15	14	13	12	11	10	9	8
Read	0			CMPA	CMPB	CMPC	CMPD	SARADC
Write								
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
Read	CYCADC	0	CRC	0	PIT0	PIT1	PDB0	PDB1
Write								
Reset	0	0	0	0	0	0	0	0

SIM_PCE2 field descriptions

Field	Description
15–13 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
12 CMPA	CMPA IPBus Clock Enable (enables both CMP and 6-bit reference DAC) 0 The peripheral is not clocked. 1 The peripheral is clocked.
11 CMPB	CMPB IPBus Clock Enable (enables both CMP and 6-bit reference DAC) 0 The peripheral is not clocked. 1 The peripheral is clocked.
10 CMPC	CMPC IPBus Clock Enable (enables both CMP and 6-bit reference DAC) 0 The peripheral is not clocked. 1 The peripheral is clocked.
9 CMPD	CMPD IPBus Clock Enable (enables both CMP and 6-bit reference DAC) 0 The peripheral is not clocked. 1 The peripheral is clocked.

Table continues on the next page...

SIM_PCE2 field descriptions (continued)

Field	Description
8 SARADC	SAR ADC IPBus Clock Enable 0 The peripheral is not clocked. 1 The peripheral is clocked.
7 CYCADC	Cyclic ADC IPBus Clock Enable 0 The peripheral is not clocked. 1 The peripheral is clocked.
6 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
5 CRC	CRC IPBus Clock Enable 0 The peripheral is not clocked. 1 The peripheral is clocked.
4 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
3 PIT0	Programmable Interval Timer IPBus Clock Enable 0 The peripheral is not clocked. 1 The peripheral is clocked.
2 PIT1	Programmable Interval Timer IPBus Clock Enable 0 The peripheral is not clocked. 1 The peripheral is clocked.
1 PDB0	Programmable Delay Block IPBus Clock Enable 0 The peripheral is not clocked. 1 The peripheral is clocked.
0 PDB1	Programmable Delay Block IPBus Clock Enable 0 The peripheral is not clocked. 1 The peripheral is clocked.

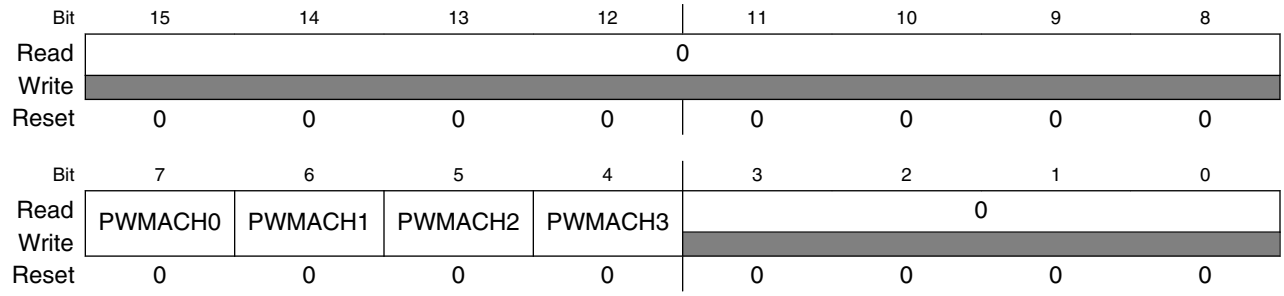
11.2.15 Peripheral Clock Enable Register 3 (SIM_PCE3)

The Peripheral Clock Enable registers enable or disable clocking of individual peripherals. Enabling the clocks of only peripherals that are in use can achieve significant power savings. When a peripheral's clock is disabled, no functionality, including I/O, is available to that peripheral.

Peripherals should not be left in an enabled or operating mode while their clocks are disabled or while their clocks are reconfigured using the controls in the SIM or OCCS module. PCE bits should therefore be changed only while the applicable peripheral is disabled. Refer to the peripheral's description for further details.

Enabled peripheral clocks still become disabled in STOP mode unless the peripheral's STOP Disable control in the SDn registers is set to 1.

Address: E400h base + Fh offset = E40Fh



SIM_PCE3 field descriptions

Field	Description
15–8 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
7 PWMACH0	PWMA Channel 0 IPBus Clock Enable 0 The peripheral is not clocked. 1 The peripheral is clocked.
6 PWMACH1	PWMA Channel 1 IPBus Clock Enable 0 The peripheral is not clocked. 1 The peripheral is clocked.
5 PWMACH2	PWMA Channel 2 IPBus Clock Enable 0 The peripheral is not clocked. 1 The peripheral is clocked.
4 PWMACH3	PWMA Channel 3 IPBus Clock Enable 0 The peripheral is not clocked. 1 The peripheral is clocked.
Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

11.2.16 STOP Disable Register 0 (SIM_SD0)

By default, peripheral clocks are disabled in stop mode to maximize power savings. The stop disable controls in the SD register override individual peripheral clocks so that they continue to operate in stop mode. Because asserting an interrupt makes the system return to run mode, this feature is provided so that selected peripherals can continue to operate in stop mode for the purpose of generating a wakeup interrupt.

For power-conscious applications, only an essential set of peripherals should be configured to remain operational in stop mode.

Peripherals should be put in a non-operating (disabled) configuration before the device enters stop mode unless their corresponding STOP Disable control is set to 1. Refer to the peripheral's description for details. IP bus reads and writes cannot be made to a module that has its clock disabled.

The SD register controls have lower priority than the PCE (Peripheral Clock Enable) register controls. If the peripheral PCE control is set to 0, the peripheral clock is disabled in all modes, including stop mode.

NOTE

The FTFL (flash) module does not have an SD bit because it does not have a distinct peripheral clock that can be gated.

Address: E400h base + 10h offset = E410h

Bit	15	14	13	12	11	10	9	8
Read Write	TA0	TA1	TA2	TA3	TB0	TB1	TB2	TB3
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
Read	0	GPIOA	GPIOB	GPIOC	GPIOD	GPIOE	GPIOF	0
Write								
Reset	0	0	0	0	0	0	0	0

SIM_SD0 field descriptions

Field	Description
15 TA0	TMRA0 IPBus STOP Disable This bit enables peripheral clocking during stop mode to the indicated peripheral provided the corresponding PCEn bit is set to 1 so that the peripheral clock itself is enabled. 0 The peripheral is not clocked in stop mode. 1 The peripheral is clocked in stop mode.
14 TA1	TMRA1 IPBus STOP Disable This bit enables peripheral clocking during stop mode to the indicated peripheral provided the corresponding PCEn bit is set to 1 so that the peripheral clock itself is enabled. 0 The peripheral is not clocked in stop mode. 1 The peripheral is clocked in stop mode.
13 TA2	TMRA2 IPBus STOP Disable This bit enables peripheral clocking during stop mode to the indicated peripheral provided the corresponding PCEn bit is set to 1 so that the peripheral clock itself is enabled. 0 The peripheral is not clocked in stop mode. 1 The peripheral is clocked in stop mode.
12 TA3	TMRA3 IPBus STOP Disable This bit enables peripheral clocking during stop mode to the indicated peripheral provided the corresponding PCEn bit is set to 1 so that the peripheral clock itself is enabled.

Table continues on the next page...

SIM_SD0 field descriptions (continued)

Field	Description
	0 The peripheral is not clocked in stop mode. 1 The peripheral is clocked in stop mode.
11 TB0	TMRB0 IPBus STOP Disable This bit enables peripheral clocking during stop mode to the indicated peripheral provided the corresponding PCEn bit is set to 1 so that the peripheral clock itself is enabled. 0 The peripheral is not clocked in stop mode. 1 The peripheral is clocked in stop mode.
10 TB1	TMRB1 IPBus STOP Disable This bit enables peripheral clocking during stop mode to the indicated peripheral provided the corresponding PCEn bit is set to 1 so that the peripheral clock itself is enabled. 0 The peripheral is not clocked in stop mode. 1 The peripheral is clocked in stop mode.
9 TB2	TMRB2 IPBus STOP Disable This bit enables peripheral clocking during stop mode to the indicated peripheral provided the corresponding PCEn bit is set to 1 so that the peripheral clock itself is enabled. 0 The peripheral is not clocked in stop mode. 1 The peripheral is clocked in stop mode.
8 TB3	TMRB3 IPBus STOP Disable This bit enables peripheral clocking during stop mode to the indicated peripheral provided the corresponding PCEn bit is set to 1 so that the peripheral clock itself is enabled. 0 The peripheral is not clocked in stop mode. 1 The peripheral is clocked in stop mode.
7 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
6 GPIOA	GPIOA IPBus STOP Disable This bit enables peripheral clocking during stop mode to the indicated peripheral provided the corresponding PCEn bit is set to 1 so that the peripheral clock itself is enabled. 0 The peripheral is not clocked in stop mode. 1 The peripheral is clocked in stop mode.
5 GPIOB	GPIOB IPBus STOP Disable This bit enables peripheral clocking during stop mode to the indicated peripheral provided the corresponding PCEn bit is set to 1 so that the peripheral clock itself is enabled. 0 The peripheral is not clocked in stop mode. 1 The peripheral is clocked in stop mode.
4 GPIOC	GPIOC IPBus STOP Disable This bit enables peripheral clocking during stop mode to the indicated peripheral provided the corresponding PCEn bit is set to 1 so that the peripheral clock itself is enabled. 0 The peripheral is not clocked in stop mode. 1 The peripheral is clocked in stop mode.

Table continues on the next page...

SIM_SD0 field descriptions (continued)

Field	Description
3 GPIOD	GPIOD IPBus STOP Disable This bit enables peripheral clocking during stop mode to the indicated peripheral provided the corresponding PCEn bit is set to 1 so that the peripheral clock itself is enabled. 0 The peripheral is not clocked in stop mode. 1 The peripheral is clocked in stop mode.
2 GPIOE	GPIOE IPBus STOP Disable This bit enables peripheral clocking during stop mode to the indicated peripheral provided the corresponding PCEn bit is set to 1 so that the peripheral clock itself is enabled. 0 The peripheral is not clocked in stop mode. 1 The peripheral is clocked in stop mode.
1 GPIOF	GPIOF IPBus STOP Disable This bit enables peripheral clocking during stop mode to the indicated peripheral provided the corresponding PCEn bit is set to 1 so that the peripheral clock itself is enabled. 0 The peripheral is not clocked in stop mode. 1 The peripheral is clocked in stop mode.
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

11.2.17 Peripheral Clock STOP Disable Register 1 (SIM_SD1)

By default, peripheral clocks are disabled in stop mode to maximize power savings. The stop disable controls in the SD register override individual peripheral clocks so that they continue to operate in stop mode. Because asserting an interrupt makes the system return to run mode, this feature is provided so that selected peripherals can continue to operate in stop mode for the purpose of generating a wakeup interrupt.

For power-conscious applications, only an essential set of peripherals should be configured to remain operational in stop mode.

Peripherals should be put in a non-operating (disabled) configuration before the device enters stop mode unless their corresponding STOP Disable control is set to 1. Refer to the peripheral's description for details. IP bus reads and writes cannot be made to a module that has its clock disabled.

The SD register controls have lower priority than the PCE (Peripheral Clock Enable) register controls. If the peripheral PCE control is set to 0, the peripheral clock is disabled in all modes, including stop mode.

Address: E400h base + 11h offset = E411h

Bit	15	14	13	12	11	10	9	8
Read	0		DAC	SCI0	SCI1	0	QSPI0	0
Write								
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
Read	0	IIC0	IIC1	0				FLEXCAN
Write								
Reset	0	0	0	0	0	0	0	0

SIM_SD1 field descriptions

Field	Description
15–14 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
13 DAC	DAC IPBus STOP Disable This bit enables peripheral clocking during stop mode to the indicated peripheral provided the corresponding PCEn bit is set to 1 so that the peripheral clock itself is enabled. 0 The peripheral is not clocked in stop mode. 1 The peripheral is clocked in stop mode.
12 SCI0	SCI0 IPBus STOP Disable This bit enables peripheral clocking during stop mode to the indicated peripheral provided the corresponding PCEn bit is set to 1 so that the peripheral clock itself is enabled. 0 The peripheral is not clocked in stop mode. 1 The peripheral is clocked in stop mode.
11 SCI1	SCI1 IPBus STOP Disable This bit enables peripheral clocking during stop mode to the indicated peripheral provided the corresponding PCEn bit is set to 1 so that the peripheral clock itself is enabled. 0 The peripheral is not clocked in stop mode. 1 The peripheral is clocked in stop mode.
10 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
9 QSPI0	QSPI0 IPBus STOP Disable This bit enables peripheral clocking during stop mode to the indicated peripheral provided the corresponding PCEn bit is set to 1 so that the peripheral clock itself is enabled. 0 The peripheral is not clocked in stop mode. 1 The peripheral is clocked in stop mode.
8 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
7 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
6 IIC0	IIC0 IPBus STOP Disable This bit enables peripheral clocking during stop mode to the indicated peripheral provided the corresponding PCEn bit is set to 1 so that the peripheral clock itself is enabled.

Table continues on the next page...

SIM_SD1 field descriptions (continued)

Field	Description
	0 The peripheral is not clocked in stop mode. 1 The peripheral is clocked in stop mode, but the IIC0 module will not enter stop mode.
5 IIC1	IIC1 IPBus STOP Disable This bit enables peripheral clocking during stop mode to the indicated peripheral provided the corresponding PCEn bit is set to 1 so that the peripheral clock itself is enabled. 0 The peripheral is not clocked in stop mode. 1 The peripheral is clocked in stop mode, but the IIC1 module will not enter stop mode.
4–1 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
0 FLEXCAN	FlexCAN IPBus STOP Disable This bit enables peripheral clocking during stop mode to the indicated peripheral provided the corresponding PCEn bit is set to 1 so that the peripheral clock itself is enabled. 0 The peripheral is not clocked in stop mode. 1 The peripheral is clocked in stop mode.

11.2.18 Peripheral Clock STOP Disable Register 2 (SIM_SD2)

By default, peripheral clocks are disabled in stop mode to maximize power savings. The stop disable controls in the SD register override individual peripheral clocks so that they continue to operate in stop mode. Because asserting an interrupt makes the system return to run mode, this feature is provided so that selected peripherals can continue to operate in stop mode for the purpose of generating a wakeup interrupt.

For power-conscious applications, only an essential set of peripherals should be configured to remain operational in stop mode.

Peripherals should be put in a non-operating (disabled) configuration before the device enters stop mode unless their corresponding STOP Disable control is set to 1. Refer to the peripheral's description for details. IP bus reads and writes cannot be made to a module that has its clock disabled.

The SD register controls have lower priority than the PCE (Peripheral Clock Enable) register controls. If the peripheral PCE control is set to 0, the peripheral clock is disabled in all modes, including stop mode.

Address: E400h base + 12h offset = E412h

Bit	15	14	13	12	11	10	9	8
Read	0			CMPA	CMPB	CMPC	CMPD	SARADC
Write								
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
Read	CYCADC	0	CRC	0	PIT0	PIT1	PDB0	PDB1
Write								
Reset	0	0	0	0	0	0	0	0

SIM_SD2 field descriptions

Field	Description
15–13 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
12 CMPA	CMPA IPBus STOP Disable (disables both CMP and 6-bit reference DAC) This bit enables peripheral clocking during stop mode to the indicated peripheral provided the corresponding PCEn bit is set to 1 so that the peripheral clock itself is enabled. 0 The peripheral is not clocked in stop mode. 1 The peripheral is clocked in stop mode.
11 CMPB	CMPB IPBus STOP Disable (disables both CMP and 6-bit reference DAC) This bit enables peripheral clocking during stop mode to the indicated peripheral provided the corresponding PCEn bit is set to 1 so that the peripheral clock itself is enabled. 0 The peripheral is not clocked in stop mode. 1 The peripheral is clocked in stop mode.
10 CMPC	CMPC IPBus STOP Disable (disables both CMP and 6-bit reference DAC) This bit enables peripheral clocking during stop mode to the indicated peripheral provided the corresponding PCEn bit is set to 1 so that the peripheral clock itself is enabled. 0 The peripheral is not clocked in stop mode. 1 The peripheral is clocked in stop mode.
9 CMPD	CMPD IPBus STOP Disable (disables both CMP and 6-bit reference DAC) This bit enables peripheral clocking during stop mode to the indicated peripheral provided the corresponding PCEn bit is set to 1 so that the peripheral clock itself is enabled. 0 The peripheral is not clocked in stop mode. 1 The peripheral is clocked in stop mode.
8 SARADC	SAR ADC IPBus STOP Disable This bit enables peripheral clocking during stop mode to the indicated peripheral provided the corresponding PCEn bit is set to 1 so that the peripheral clock itself is enabled. 0 The peripheral is not clocked in stop mode. 1 The peripheral is clocked in stop mode.
7 CYCADC	Cyclic ADC IPBus STOP Disable This bit enables peripheral clocking during stop mode to the indicated peripheral provided the corresponding PCEn bit is set to 1 so that the peripheral clock itself is enabled. 0 The peripheral is not clocked in stop mode. 1 The peripheral is clocked in stop mode.
6 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

Table continues on the next page...

SIM_SD2 field descriptions (continued)

Field	Description
5 CRC	CRC IPBus STOP Disable This bit enables peripheral clocking during stop mode to the indicated peripheral provided the corresponding PCEn bit is set to 1 so that the peripheral clock itself is enabled. 0 The peripheral is not clocked in stop mode. 1 The peripheral is clocked in stop mode.
4 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
3 PIT0	Programmable Interval Timer IPBus STOP Disable This bit enables peripheral clocking during stop mode to the indicated peripheral provided the corresponding PCEn bit is set to 1 so that the peripheral clock itself is enabled. 0 The peripheral is not clocked in stop mode. 1 The peripheral is clocked in stop mode.
2 PIT1	Programmable Interval Timer IPBus STOP Disable This bit enables peripheral clocking during stop mode to the indicated peripheral provided the corresponding PCEn bit is set to 1 so that the peripheral clock itself is enabled. 0 The peripheral is not clocked in stop mode. 1 The peripheral is clocked in stop mode.
1 PDB0	Programmable Delay Block IPBus STOP Disable This bit enables peripheral clocking during stop mode to the indicated peripheral provided the corresponding PCEn bit is set to 1 so that the peripheral clock itself is enabled. 0 The peripheral is not clocked in stop mode. 1 The peripheral is clocked in stop mode.
0 PDB1	Programmable Delay Block IPBus STOP Disable This bit enables peripheral clocking during stop mode to the indicated peripheral provided the corresponding PCEn bit is set to 1 so that the peripheral clock itself is enabled. 0 The peripheral is not clocked in stop mode. 1 The peripheral is clocked in stop mode.

11.2.19 Peripheral Clock STOP Disable Register 3 (SIM_SD3)

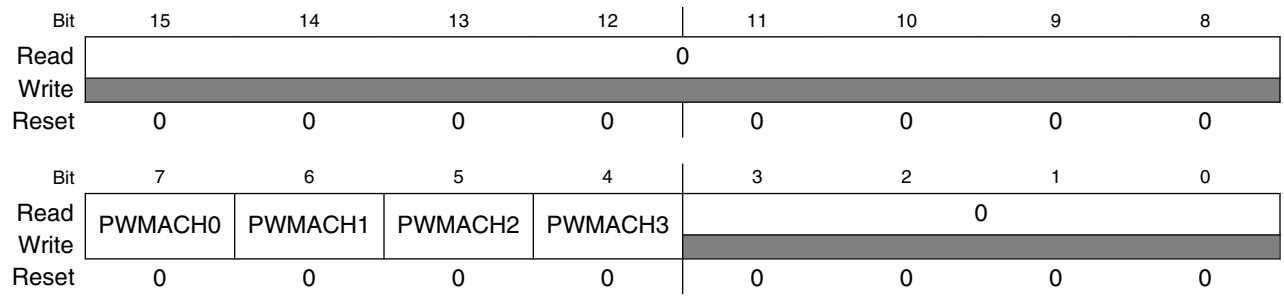
By default, peripheral clocks are disabled in stop mode to maximize power savings. The stop disable controls in the SD register override individual peripheral clocks so that they continue to operate in stop mode. Because asserting an interrupt makes the system return to run mode, this feature is provided so that selected peripherals can continue to operate in stop mode for the purpose of generating a wakeup interrupt.

For power-conscious applications, only an essential set of peripherals should be configured to remain operational in stop mode.

Peripherals should be put in a non-operating (disabled) configuration before the device enters stop mode unless their corresponding STOP Disable control is set to 1. Refer to the peripheral's description for details. IP bus reads and writes cannot be made to a module that has its clock disabled.

The SD register controls have lower priority than the PCE (Peripheral Clock Enable) register controls. If the peripheral PCE control is set to 0, the peripheral clock is disabled in all modes, including stop mode.

Address: E400h base + 13h offset = E413h



SIM_SD3 field descriptions

Field	Description
15–8 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
7 PWMACH0	PWMA Channel 0 IPBus STOP Disable This bit enables peripheral clocking during stop mode to the indicated peripheral provided the corresponding PCEn bit is set to 1 so that the peripheral clock itself is enabled. 0 The peripheral is not clocked in stop mode. 1 The peripheral is clocked in stop mode.
6 PWMACH1	PWMA Channel 1 IPBus STOP Disable This bit enables peripheral clocking during stop mode to the indicated peripheral provided the corresponding PCEn bit is set to 1 so that the peripheral clock itself is enabled. 0 The peripheral is not clocked in stop mode. 1 The peripheral is clocked in stop mode.
5 PWMACH2	PWMA Channel 2 IPBus STOP Disable This bit enables peripheral clocking during stop mode to the indicated peripheral provided the corresponding PCEn bit is set to 1 so that the peripheral clock itself is enabled. 0 The peripheral is not clocked in stop mode. 1 The peripheral is clocked in stop mode.
4 PWMACH3	PWMA Channel 3 IPBus STOP Disable This bit enables peripheral clocking during stop mode to the indicated peripheral provided the corresponding PCEn bit is set to 1 so that the peripheral clock itself is enabled. 0 The peripheral is not clocked in stop mode. 1 The peripheral is clocked in stop mode.

Table continues on the next page...

SIM_SD3 field descriptions (continued)

Field	Description
Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

11.2.20 I/O Short Address Location Register (SIM_IOSAHI)

The I/O short address location registers specify the memory referenced through the I/O short address mode, which allows the instruction to specify the lower 6 bits of the address. The upper address bits are not directly controllable. The I/O short address location registers allow limited control of the full address.

With this register set, an interrupt driver can set the combined ISAL register to point to its peripheral registers and then use the I/O short addressing mode to reference them. The ISR should restore this register to its previous contents prior to returning from interrupt.

NOTE

The default value of this register points to the beginning of the slave peripheral region at E000h.

NOTE

The pipeline delay between setting this register set and using short I/O addressing with the new value is five cycles.

Address: E400h base + 14h offset = E414h

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	0														ISAL[23:22]	
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SIM_IOSAHI field descriptions

Field	Description
15–2 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
ISAL[23:22]	Bits 23:22 of the address The I/O short address is calculated by concatenating the combined ISAL value with the 6-bit short address from the CPU short address opcode.

11.2.21 I/O Short Address Location Register (SIM_IOSALO)

The I/O short address location registers specify the memory referenced through the I/O short address mode, which allows the instruction to specify the lower 6 bits of the address. The upper address bits are not directly controllable. The I/O short address location registers allow limited control of the full address.

With this register set, an interrupt driver can set the combined ISAL register to point to its peripheral registers and then use the I/O short addressing mode to reference them. The ISR should restore this register to its previous contents prior to returning from interrupt.

NOTE

The default value of this register points to the beginning of the slave peripheral region at E000h.

NOTE

The pipeline delay between setting this register set and using short I/O addressing with the new value is five cycles.

Address: E400h base + 15h offset = E415h

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read Write	ISAL[21:6]															
Reset	0	0	0	0	0	0	1	1	1	0	0	0	0	0	0	0

SIM_IOSALO field descriptions

Field	Description
ISAL[21:6]	Bits 21:6 of the address The I/O short address is calculated by concatenating the combined ISAL value with the 6-bit short address from the CPU short address opcode.

11.2.22 Protection Register (SIM_PROT)

This register provides write protection of selected control fields for safety critical applications. The primary purpose is to prevent unsafe conditions due to the unintentional modification of these fields between the onset of a code runaway and a reset by the COP watchdog. GPIO and Internal Peripheral Select Protection (GIPSP) write protects the registers in the SIM, XBAR, AOI, and GPIO modules that control inter-peripheral signal multiplexing and I/O cell configuration. Peripheral Clock Enable Protection (PCEP) write protects the SIM registers that contain peripheral-specific clock controls. GDP provides write protection of the GPIO Port D registers separately from the other ports protected by the GIPSP field. Some peripherals provide additional safety features. Refer to peripheral descriptions for details.

GIPSP protects the contents of the SIM registers that control multiplexing of inter-peripheral connections (GPSn and IPSn) as well as all inter-peripheral XBAR and AOI registers. GIPSP also write protects some registers in the GPIO module other than for port D, including the GPIOn_PER registers that select between peripheral and GPIO ownership of the I/O cell, the GPIOn_PPMODE registers the control the I/O cell's push/pull mode, and GPIOn_DRIVE registers that control the I/O cell's drive strength.

GDP provides write protection for the GPIO Port D registers, which include JTAG and reset functionality. These include GPIO_D_PER, GPIO_D_PPMODE, and GPIO_D_DRIVE as well as the GPSDL register.

PCEP write protects the SIM peripheral clock enable registers (PCEn), the SIM peripheral stop disable registers (SDn), the SIM peripheral software reset registers (PSWRn), and the SIM peripheral clock rate registers (PCR).

For flexibility, write protection control values may themselves be optionally locked (write protected). To this end, protection controls in this register have two bit values. The right bit determines the setting of the control, and the left bit determines whether the value is locked. While a protection control remains unlocked, protection can be disabled and re-enabled as desired. When a protection control is locked, its value can be altered only by a device reset, which restores its default non-locked value.

Address: E400h base + 16h offset = E416h

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	0								PMODE		GDP		PCEP		GIPSP	
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SIM_PROT field descriptions

Field	Description
15–8 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
7–6 PMODE	Power Mode Control Write Protection Enables write protection of the PWRMODE register. 00 Write protection off (default). 01 Write protection on. 10 Write protection off and locked until chip reset. 11 Write protection on and locked until chip reset.
5–4 GDP	GPIO Port D Protection Enables write protection of GPIO_D_PER, GPIO_D_PPMODE, and GPIO_D_DRIVE registers. This field also write protects the GPSDL register and CTRL[RST_FILT] bit. GPIO Port D contains JTAG and reset functions that are protected separately from other GPIO ports. 00 Write protection off (default). 01 Write protection on. 10 Write protection off and locked until chip reset. 11 Write protection on and locked until chip reset.
3–2 PCEP	Peripheral Clock Enable Protection Enables write protection of all fields in the PCEn, SDn, PSWRn, and PCR registers. 00 Write protection off (default). 01 Write protection on. 10 Write protection off and locked until chip reset. 11 Write protection on and locked until chip reset.
GIPSP	GPIO and Internal Peripheral Select Protection Enables write protection of GPSn and IPSn registers in the SIM. This field also write protects registers in other modules including all XBAR, AOI, GPIOn_PER, GPIOn_PPMODE, and GPIOn_DRIVE registers. This field does not protect GPIO Port D registers or the GPSDL register; the GDP field provides that protection. 00 Write protection off (default). 01 Write protection on. 10 Write protection off and locked until chip reset. 11 Write protection on and locked until chip reset.

11.2.23 GPIOA LSBs Peripheral Select Register (SIM_GPSAL)

Most I/O pads have an associated GPIO function that, when enabled, can control and observe the I/O pad. As an alternative to the GPIO function, most I/O can be configured to connect to one of several internal peripheral functions. When the GPIOn_PER bit for an I/O is set to 0, the associated GPIO has control of the I/O to the exclusion of any internal peripheral function. The GPIOn_PER bit for an I/O must be set to 1 for the I/O to access any of its internal peripheral functions. If an I/O pad can be configured to connect to one of several peripheral functions, the GPSn field for that GPIO is used to select the active peripheral function.

NOTE

A GPIO with only one peripheral function does not require a GPS field. That peripheral function is always enabled when the PER field of the corresponding GPIO is set to 1.

In some cases there are multiple I/O pads, each of which can be configured via their GPIOn_PER and SIM GPS settings to connect to the same internal peripheral function. If more than one I/O is connected to the same peripheral output function, the peripheral output signal safely fans out to each of these I/O. However, if more than one I/O is connected to the same peripheral input signal, that peripheral input is the logical OR or AND of these multiple sources and is therefore invalid. As a result, at most one I/O should be configured to control a specific peripheral input function.

The user may opt not to use a specific peripheral input or output function. If no I/O is configured to observe a peripheral output function, then the peripheral output signal is inaccessible. If no I/O is configured to control a peripheral input signal, that peripheral input is tied to an application-appropriate constant value.

When a GPIO's internal peripheral functions include a connection to an output of an Internal Crossbar Switch (XBAR) module, that GPIO can be driven by any input to the XBAR by properly configuring the XBAR module. Similarly, when a GPIO's internal functions include a connection to an input of an Internal Crossbar Switch (XBAR) module, that GPIO can feed any device fed by the XBAR's outputs by properly configuring the XBAR module.

GPSn settings should not be altered while an affected peripheral is in an enabled (operational) configuration. See the peripheral's description for further details.

Address: E400h base + 17h offset = E417h

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	0															A0
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SIM_GPSAL field descriptions

Field	Description
15–1 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
0 A0	Configure GPIO A0 0 Function = ANA0/CMPA3; Peripheral = ADC/CMPA; Direction = AN_IN 1 Function = CMPC_O; Peripheral = CMPC; Direction = OUT

11.2.24 GPIOC LSBs Peripheral Select Register (SIM_GPSCL)

See the description of the GPSAL register.

Address: E400h base + 19h offset = E419h

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	0	C7	C6		0	C5	C4		C3		C2		0			C0
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SIM_GPSCL field descriptions

Field	Description
15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
14 C7	Configure GPIO C7 0 Function = SS0_B; Peripheral = SPI0; Direction = IO 1 Function = TXD0; Peripheral = SCI0; Direction = IO
13–12 C6	Configure GPIO C6 00 Function = TA2; Peripheral = TMRA; Direction = IO 01 Function = XB_IN3; Peripheral = XBAR; Direction = IN 10 Function = CMPREF; Peripheral = HSCMP A/B/C/D; Direction = AN_IN 11 Reserved
11 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
10 C5	Configure GPIO C5

Table continues on the next page...

SIM_GPSCL field descriptions (continued)

Field	Description
	0 Function = DACO; Peripheral = DAC; Direction = AN_OUT 1 Function = XB_IN7; Peripheral = XBAR; Direction = IN
9–8 C4	Configure GPIO C4 00 Function = TA1; Peripheral = TMRA; Direction = IO 01 Function = CMPB_O; Peripheral = CMPB; Direction = OUT 10 Function = XB_IN8; Peripheral = XBAR; Direction = IN 11 Function = EWM_OUT_B; Peripheral = EWM; Direction = OUT
7–6 C3	Configure GPIO C3 00 Function = TA0; Peripheral = TMRA; Direction = IO 01 Function = CMPA_O; Peripheral = CMPA; Direction = OUT 10 Function = RXD0; Peripheral = SCI0; Direction = IN 11 Function = CLKIN1; Peripheral = SIM; Direction = IN
5–4 C2	Configure GPIO C2 00 Function = TXD0; Peripheral = SCI0; Direction = IO 01 Function = TB0; Peripheral = TMRB; Direction = IO 10 Function = XB_IN2; Peripheral = XBAR; Direction = IN 11 Function = CLKOUT0; Peripheral = SIM; Direction = OUT
3–1 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
0 C0	Configure GPIO C0 0 Function = EXTAL; Peripheral = OSC; Direction = AN_IN 1 Function = CLKIN; Peripheral = OCCS; Direction = IN

11.2.25 GPIOC MSBs Peripheral Select Register (SIM_GPSCH)

See the description of the GPSAL register.

Address: E400h base + 1Ah offset = E41Ah

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	0	C15	0	C14	C13		C12		C11		C10		0	C9	C8	
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SIM_GPSCH field descriptions

Field	Description
15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

Table continues on the next page...

SIM_GPSCH field descriptions (continued)

Field	Description
14 C15	Configure GPIO C15 0 Function = SCL0; Peripheral = IIC0; Direction = OD_IO 1 Function = XB_OUT5; Peripheral = XBAR; Direction = OUT
13 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
12 C14	Configure GPIO C14 0 Function = SDA0; Peripheral = IIC0; Direction = OD_IO 1 Function = XB_OUT4; Peripheral = XBAR; Direction = OUT
11–10 C13	Configure GPIO C13 00 Function = TA3; Peripheral = TMRA; Direction = IO 01 Function = XB_IN6; Peripheral = XBAR; Direction = IN 10 Function = EWM_OUT_B; Peripheral = EWM; Direction = OUT 11 Reserved
9–8 C12	Configure GPIO C12 00 Function = CANRX; Peripheral = FlexCAN; Direction = IN 01 Function = SDA1; Peripheral = IIC1; Direction = OD_IO 10 Function = RXD1; Peripheral = SCI1; Direction = IN 11 Reserved
7–6 C11	Configure GPIO C11 00 Function = CANTX; Peripheral = FlexCAN; Direction = OD_OUT 01 Function = SCL1; Peripheral = IIC1; Direction = OD_IO 10 Function = TXD1; Peripheral = SCI1; Direction = IO 11 Reserved
5–4 C10	Configure GPIO C10 00 Function = MOSI0; Peripheral = SPI0; Direction = IO 01 Function = XB_IN5; Peripheral = XBAR; Direction = IN 10 Function = MISO0; Peripheral = SPI0; Direction = IO 11 Reserved
3 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
2 C9	Configure GPIO C9 0 Function = SCLK0; Peripheral = SPI0; Direction = IO 1 Function = XB_IN4; Peripheral = XBAR; Direction = IN
C8	Configure GPIO C8 00 Function = MISO0; Peripheral = SPI0; Direction = IO 01 Function = RXD0; Peripheral = SCI0; Direction = IN 10 Function = XB_IN9; Peripheral = XBAR; Direction = IN 11 Reserved

11.2.26 GPIOE LSBs Peripheral Select Register (SIM_GPSEL)

See the description of the GPSAL register.

Address: E400h base + 1Ch offset = E41Ch

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	E7		E6		0	E5	0	E4	0							
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SIM_GPSEL field descriptions

Field	Description
15–14 E7	Configure GPIO E7 00 Function = PWMA_3A; Peripheral = PWMA; Direction = IO 01 Function = XB_IN5; Peripheral = XBAR; Direction = IN 10 Reserved 11 Reserved
13–12 E6	Configure GPIO E6 00 Function = PWMA_3B; Peripheral = PWMA; Direction = IO 01 Function = XB_IN4; Peripheral = XBAR; Direction = IN 10 Reserved 11 Reserved
11 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
10 E5	Configure GPIO E5 0 Function = PWMA_2A; Peripheral = PWMA; Direction = IO 1 Function = XB_IN3; Peripheral = XBAR; Direction = IN
9 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
8 E4	Configure GPIO E4 0 Function = PWMA_2B; Peripheral = PWMA; Direction = IO 1 Function = XB_IN2; Peripheral = XBAR; Direction = IN
Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

11.2.27 GPIOF LSBs Peripheral Select Register (SIM_GPSFL)

See the description of the GPSAL register.

Address: E400h base + 1Eh offset = E41Eh

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	F7		F6		0	F5	0	F4	0	F3	0	F2	F1		F0	
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SIM_GPSFL field descriptions

Field	Description
15–14 F7	Configure GPIO F7 00 Function = TB3; Peripheral = TMRB; Direction = IO 01 Function = CMPC_O; Peripheral = HSCMPC; Direction = OUT 10 Function = SS1_B; Peripheral = SPI1; Direction = IO 11 Function = XB_IN3; Peripheral = XBAR; Direction = IN
13–12 F6	Configure GPIO F6 00 Function = TB2; Peripheral = TMRB; Direction = IO 01 Function = PWMA_3X; Peripheral = PWMA; Direction = IO 10 Reserved 11 Function = XB_IN2; Peripheral = XBAR; Direction = IN
11 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
10 F5	Configure GPIO F5 0 Function = RXD1; Peripheral = SCI1; Direction = IN 1 Function = XB_OUT9; Peripheral = XBAR; Direction = OUT
9 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
8 F4	Configure GPIO F4 0 Function = TXD1; Peripheral = SCI1; Direction = IO 1 Function = XB_OUT8; Peripheral = XBAR; Direction = OUT
7 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
6 F3	Configure GPIO F3 0 Function = SDA1; Peripheral = IIC1; Direction = OD_IO 1 Function = XB_OUT7; Peripheral = XBAR; Direction = OUT
5 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
4 F2	Configure GPIO F2

Table continues on the next page...

SIM_GPSFL field descriptions (continued)

Field	Description
	0 Function = SCL1; Peripheral = IIC1; Direction = OD_IO 1 Function = XB_OUT6; Peripheral = XBAR; Direction = OUT
3–2 F1	Configure GPIO F1 00 Function = CLKOUT1; Peripheral = SIM; Direction = OUT 01 Function = XB_IN7; Peripheral = XBAR; Direction = IN 10 Function = CMPD_O; Peripheral = HSCMPD; Direction = OUT 11 Reserved
F0	Configure GPIO F0 00 Function = XB_IN6; Peripheral = XBAR; Direction = IN 01 Function = TB2; Peripheral = TMRB; Direction = IO 10 Function = SCLK1; Peripheral = SPI1; Direction = IO 11 Reserved

11.2.28 GPIOF MSBs Peripheral Select Register (SIM_GPSFH)

See the description of the GPSAL register.

Address: E400h base + 1Fh offset = E41Fh

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	0														F8	
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SIM_GPSFH field descriptions

Field	Description
15–2 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
F8	Configure GPIO F8 00 Function = RXD0; Peripheral = SCI0; Direction = IN 01 Function =TB1; Peripheral = TMRB; Direction = IO 10 Function = CMPD_O; Peripheral = HSCMPD; Direction = OUT 11 Reserved

11.2.29 Internal Peripheral Select Register 0 (SIM_IPS0)

The IPS register implements controls that affect the integration or communication between parts. Various circumstances require these controls.

Some peripheral inputs have the ability to be fed either by XBAR outputs or by GPIO. In these cases, an additional layer of internal multiplexing selects which source is active and feeds the peripheral input.

In other cases, peripherals have control relationships. For example, one peripheral may be configured to operate as a slave of another and require inputs to be configured appropriately based on that choice. This configuration may require the setting of control inputs or the switching of muxing relationships between the blocks.

NOTE

The TMRBn and TMRAn fields select which signal drives the corresponding TMRBn and TMRAn inputs. The choice in each case is between one or more external I/O pads and an XBAR output. When selecting the external I/O, you must set the GPIOn_PER bit for that I/O to 1 and configure that GPIO's SIM GPS field (if there is one) to feed that TMR channel. A standard requirement of GPS muxing to avoid contention is to configure, at most, one GPIO at a time to feed a specific peripheral input function. When more than one GPIO source is listed, the muxing uses the signal from the GPIO input that is currently sourcing the signal.

NOTE

The PWMAFn fields select which signal drives the corresponding PWMA Fault n inputs. The choice in each case is between an external IO pad and an XBAR output. When selecting the external I/O, set the GPIOn_PER bit for that I/O to 1 and configure that GPIO's SIM GPS field (if there is one) to feed that PWM Fault signal.

Address: E400h base + 22h offset = E422h

Bit	15	14	13	12	11	10	9	8
Read Write	TB3	TB2	TB1	TB0	TA3	TA2	TA1	TA0
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
Read	0	0			PWMAF3	PWMAF2	PWMAF1	PWMAF0
Write								
Reset	0	0	0	0	0	0	0	0

SIM_IPS0 field descriptions

Field	Description
15 TB3	Select TMRB3 Input

Table continues on the next page...

SIM_IPS0 field descriptions (continued)

Field	Description
	0 Function = GPIOF7; Peripheral = GPIOF; Direction = IN 1 Function = XB_OUT37; Peripheral = XBAR; Direction = IN
14 TB2	Select TMRB2 Input 0 Function = GPIOF6, GPIOF0; Peripheral = GPIOF; Direction = IN 1 Function = XB_OUT36; Peripheral = XBAR; Direction = IN
13 TB1	Select TMRB1 Input 0 Function = GPIOF8; Peripheral = GPIOF; Direction = IN 1 Function = XB_OUT35; Peripheral = XBAR; Direction = IN
12 TB0	Select TMRB0 Input 0 Function = GPIOC2; Peripheral = GPIOC; Direction = IN 1 Function = XB_OUT34; Peripheral = XBAR; Direction = IN
11 TA3	Select TMRA3 Input 0 Function = GPIOC13; Peripheral = GPIOC; Direction = IN 1 Function = XB_OUT52; Peripheral = XBAR; Direction = IN
10 TA2	Select TMRA2 Input 0 Function = GPIOC6; Peripheral = GPIOC; Direction = IN 1 Function = XB_OUT51; Peripheral = XBAR; Direction = IN
9 TA1	Select TMRA1 Input 0 Function = GPIOC4; Peripheral = GPIOC; Direction = IN 1 Function = XB_OUT50; Peripheral = XBAR; Direction = IN
8 TA0	Select TMRA0 Input 0 Function = GPIOC3; Peripheral = GPIOC; Direction = IN 1 Function = XB_OUT49; Peripheral = XBAR; Direction = IN
7 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
6–4 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
3 PWMAF3	Select PWMA Fault 3 Input 0 Reserved 1 Function = XB_OUT32; Peripheral = XBAR; Direction = IN
2 PWMAF2	Select PWMA Fault 2 Input 0 Reserved 1 Function = XB_OUT31; Peripheral = XBAR; Direction = IN
1 PWMAF1	Select PWMA Fault 1 Input 0 Reserved 1 Function = XB_OUT30; Peripheral = XBAR; Direction = IN
0 PWMAF0	Select PWMA Fault 0 Input

Table continues on the next page...

SIM_IPS0 field descriptions (continued)

Field	Description
	0 Reserved 1 Function = XB_OUT29; Peripheral = XBAR; Direction = IN

11.2.30 Miscellaneous Register 0 (SIM_MISC0)

This register controls various integration features of the chip.

Address: E400h base + 23h offset = E423h

Bit	15	14	13	12	11	10	9	8
Read	0							
Write								
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
Read	0						CLKINSEL	PIT_MSTR
Write								
Reset	0	0	0	0	0	0	0	0

SIM_MISC0 field descriptions

Field	Description
15–2 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1 CLKINSEL	CLKIN Select This bit determines the GPIO port for the CLKIN input to the OCCS. 0 CLKIN0 (GPIOC0 alt1) is selected as CLKIN 1 CLKIN1 (GPIOC3 alt3) is selected as CLKIN
0 PIT_MSTR	Select Master Programmable Interval Timer (PIT) 0 PIT0 is master PIT and PIT1 is slave PIT 1 PIT1 is master PIT and PIT0 is slave PIT

11.2.31 Peripheral Software Reset Register 0 (SIM_PSWR0)

The Peripheral Software Reset registers are used to reset individual peripherals without resetting the entire chip.

Address: E400h base + 24h offset = E424h

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	TA	0			TB	0			GPIO	0						
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SIM_PSWR0 field descriptions

Field	Description
15 TA	TMRA Software Reset This bit causes a reset of the indicated peripheral. 0 The corresponding peripheral is not reset. 1 The corresponding peripheral is reset.
14–12 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
11 TB	TMRB Software Reset This bit causes a reset of the indicated peripheral. 0 The corresponding peripheral is not reset. 1 The corresponding peripheral is reset.
10–7 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
6 GPIO	GPIO Software Reset This bit causes a reset of the indicated peripheral. 0 The corresponding peripheral is not reset. 1 The corresponding peripheral is reset.
Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

11.2.32 Peripheral Software Reset Register 1 (SIM_PSWR1)

The Peripheral Software Reset registers are used to reset individual peripherals without resetting the entire chip.

Address: E400h base + 25h offset = E425h

Bit	15	14	13	12	11	10	9	8
Read	0		DAC	SCI0	SCI1	0	QSPI0	0
Write								
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
Read	0	IIC0	IIC1	0				FLEXCAN
Write								
Reset	0	0	0	0	0	0	0	0

SIM_PSWR1 field descriptions

Field	Description
15–14 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
13 DAC	DAC Software Reset This bit causes a reset of the indicated peripheral. 0 The corresponding peripheral is not reset. 1 The corresponding peripheral is reset.
12 SCI0	SCI0 Software Reset This bit causes a reset of the indicated peripheral. 0 The corresponding peripheral is not reset. 1 The corresponding peripheral is reset.
11 SCI1	SCI1 Software Reset This bit causes a reset of the indicated peripheral. 0 The corresponding peripheral is not reset. 1 The corresponding peripheral is reset.
10 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
9 QSPI0	QSPI0 Software Reset This bit causes a reset of the indicated peripheral. 0 The corresponding peripheral is not reset. 1 The corresponding peripheral is reset.

Table continues on the next page...

SIM_PSWR1 field descriptions (continued)

Field	Description
8 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
7 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
6 IIC0	IIC0 Software Reset This bit causes a reset of the indicated peripheral. 0 The corresponding peripheral is not reset. 1 The corresponding peripheral is reset.
5 IIC1	IIC1 Software Reset This bit causes a reset of the indicated peripheral. 0 The corresponding peripheral is not reset. 1 The corresponding peripheral is reset.
4–1 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
0 FLEXCAN	FlexCAN Software Reset This bit causes a reset of the indicated peripheral. **NOTE:** This bit for the FlexCAN module is automatically cleared after 3 clock cycles. 0 The corresponding peripheral is not reset. 1 The corresponding peripheral is reset.

11.2.33 Peripheral Software Reset Register 2 (SIM_PSWR2)

The Peripheral Software Reset registers are used to reset individual peripherals without resetting the entire chip.

Address: E400h base + 26h offset = E426h

Bit	15	14	13	12	11	10	9	8
Read	EWM	0		CMP	0			SARADC
Write								
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
Read	CYCADC	0	CRC	0	PIT0	PIT1	PDB0	PDB1
Write								
Reset	0	0	0	0	0	0	0	0

SIM_PSWR2 field descriptions

Field	Description
15 EWM	EWM Software Reset This bit causes a reset of the indicated peripheral. 0 The corresponding peripheral is not reset. 1 The corresponding peripheral is reset.
14–13 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
12 CMP	CMP Software Reset This bit causes a reset of the indicated peripheral. 0 The corresponding peripheral is not reset. 1 The corresponding peripheral is reset.
11–9 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
8 SARADC	SAR ADC Software Reset This bit causes a reset of the indicated peripheral. 0 The corresponding peripheral is not reset. 1 The corresponding peripheral is reset.
7 CYCADC	Cyclic ADC Software Reset This bit causes a reset of the indicated peripheral. 0 The corresponding peripheral is not reset. 1 The corresponding peripheral is reset.
6 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
5 CRC	CRC Software Reset This bit causes a reset of the indicated peripheral. 0 The corresponding peripheral is not reset. 1 The corresponding peripheral is reset.
4 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
3 PIT0	Programmable Interval Timer Software Reset This bit causes a reset of the indicated peripheral. 0 The corresponding peripheral is not reset. 1 The corresponding peripheral is reset.
2 PIT1	Programmable Interval Timer Software Reset This bit causes a reset of the indicated peripheral. 0 The corresponding peripheral is not reset. 1 The corresponding peripheral is reset.

Table continues on the next page...

SIM_PSWR2 field descriptions (continued)

Field	Description
1 PDB0	Programmable Delay Block Software Reset This bit causes a reset of the indicated peripheral. 0 The corresponding peripheral is not reset. 1 The corresponding peripheral is reset.
0 PDB1	Programmable Delay Block Software Reset This bit causes a reset of the indicated peripheral. 0 The corresponding peripheral is not reset. 1 The corresponding peripheral is reset.

11.2.34 Peripheral Software Reset Register 3 (SIM_PSWR3)

The Peripheral Software Reset registers are used to reset individual peripherals without resetting the entire chip.

Address: E400h base + 27h offset = E427h

Bit	15	14	13	12	11	10	9	8
Read	0							
Write								
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
Read	PWMA	0			0	0		
Write								
Reset	0	0	0	0	0	0	0	0

SIM_PSWR3 field descriptions

Field	Description
15–8 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
7 PWMA	PWMA Software Reset This bit causes a reset of the indicated peripheral. 0 The corresponding peripheral is not reset. 1 The corresponding peripheral is reset.
6–4 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
3 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

Table continues on the next page...

SIM_PSWR3 field descriptions (continued)

Field	Description
Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

11.2.35 Power Mode Register (SIM_PWRMODE)

This register controls various low power modes of the chip if the FTFL module's FOPT[0] bit is set (advanced power mode is enabled). All control bits in this register are write protected. See the power mode chapter for mode details.

Address: E400h base + 28h offset = E428h

Bit	15	14	13	12	11	10	9	8
Read	0						LPMS	VLPMS
Write								
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
Read	0						LPMODE	VLPMODE
Write								
Reset	0	0	0	0	0	0	0	0

SIM_PWRMODE field descriptions

Field	Description
15–10 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
9 LPMS	LPMODE Status Indicator This status bit indicates whether the chip is in LPMODE. 0 Not in LPMODE 1 In LPMODE
8 VLPMS	VLPMODE Status Indicator This status bit indicates whether the chip is in VLPMODE. **NOTE:** This bit, along with the PMC's STS[SR27] bit, indicates an exit from VLPMODE. 0 Not in VLPMODE 1 In VLPMODE
7–2 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

Table continues on the next page...

SIM_PWRMODE field descriptions (continued)

Field	Description
1 LPMODE	LPMODE Entry/Exit Writing to this bit causes the device to enter LPMODE. To exit LPMODE, clear this bit. This bit can be written only if write protection is disabled (PROT[6] is 0). If both the LPMODE and VLPMODE bits are set, the VLPMODE bit has higher priority. 0 Start exit from LPMODE 1 Start entry to LPMODE
0 VLPMODE	VLPMODE Entry/Exit Writing to this bit causes the device to enter VLPMODE. To exit VLPMODE, clear this bit. This bit can be written only if write protection is disabled (PROT[6] is 0). If both the LPMODE and VLPMODE bits are set, the VLPMODE bit has higher priority. 0 Start exit from VLPMODE 1 Start entry to VLPMODE

11.2.36 Non-Volatile Memory Option Register 2 (High) (SIM_NVMOPT2H)

Address: E400h base + 2Ch offset = E42Ch

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	0		ROSC_8M_TTRIM						ROSC_8M_FTRIM							
Write																
Reset	0	0	0	0	0	0	1	0	0	0	1	0	0	0	0	0

SIM_NVMOPT2H field descriptions

Field	Description
15–14 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
13–10 ROSC_8M_ TTRIM	8 MHz Relaxation Oscillator Temperature Trim Value determined by factory
ROSC_8M_ FTRIM	8 MHz Relaxation Oscillator Frequency Trim Value determined by factory

11.2.37 Non-Volatile Memory Option Register 2 (Low) (SIM_NVMOPT2L)

Address: E400h base + 2Dh offset = E42Dh

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	PMC_BGTRIM				0				ROSC_32K_FTRIM							
Write																
Reset	0	1	1	1	0	0	0	1	0	0	0	0	0	0	0	0

SIM_NVMOPT2L field descriptions

Field	Description
15–12 PMC_BGTRIM	PMC Bandgap Trim Value determined by factory
11–9 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
ROSC_32K_ FTRIM	32 kHz Relaxation Oscillator Frequency Trim Value determined by factory

11.2.38 Non-Volatile Memory Option Register 3 (High) (SIM_NVMOPT3H)

Address: E400h base + 2Eh offset = E42Eh

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	SAR_VREFBG_TRIM															
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SIM_NVMOPT3H field descriptions

Field	Description
SAR_VREFBG_ TRIM	SAR ADC Bandgap Reference Capture Value determined by factory

11.3 Functional Description

11.3.1 Clock Generation Overview

The SIM uses a master clock from the OCCS module (MSTR_2X) to produce the peripheral and system (core and memory) clocks. This MSTR_2X clock input from OCCS operates at two times the system and peripheral bus rate. Peripheral and system clocks, with the exception of the RAM system clock, are generated by dividing the MSTR_2X clock by two and gating it with appropriate power mode and clock gating controls. The RAM requires a 2x system rate clock. This clock is therefore generated by gating the MSTR_2X clock with appropriate power mode and clock gating controls.

The OCCS supports several methods for deriving MSTR_2X. A master clock source (MSTR_OSC) is selected from several available sources, including an up to 50 MHz external clock input (CLKIN), a 4 MHz to 16 MHz crystal oscillator, a 32 kHz internal oscillator, or an 8 MHz / 400 kHz internal relaxation oscillator.

A high-speed clock source can be derived by multiplying MSTR_OSC frequencies between 8 MHz and 50 MHz through a PLL to a maximum allowed frequency of 400 MHz. Duty cycle correction for the high-speed clock source is achieved by a DIV2 circuit.

MSTR_2X is derived by selecting either MSTR_OSC or the high-speed clock source as the active clock source and optionally dividing it through a post-scaler. The post-scaler will support division by up to 50. As a result, the clock system has the flexibility to generate diverse MSTR_2X frequencies from 200 MHz down to below 1 Hz.

While deriving the system and peripheral clocks from MSTR_2X, the SIM provides several methods for clock gating to manage and reduce the overall power consumption of the part. These methods include low-power modes RUN/STOP/WAIT and various clock enables (PCEn registers, SDn registers, CLK_DIS, ONCE_EBL). System clocks including the core clock normally operate only in RUN mode. Peripheral clocks operate in RUN or WAIT modes when enabled by their individual clock gating controls in the SIM's PCE registers. Peripheral clocks may be individually overridden to operate in STOP mode using the SIM's SD registers—to operate select peripherals used for recovery from STOP mode back to RUN mode.

11.3.2 Power-Down Modes Overview

The core operates in one of following power-down modes.

Table 11-1. Clock Operation In Power Down Modes

Mode	System Clocks	Peripheral Clocks	Description
RUN	Core and memory clocks enabled	Peripheral clocks enabled	Device is fully functional
WAIT	Core and memory clocks disabled	Peripheral clocks enabled	Core executes WAIT instruction to enter this mode. Wait mode is typically used for power conscious applications. Possible recoveries from WAIT mode to RUN mode are: 1. Any interrupt. 2. Executing a debug mode entry command using the core JTAG interface. 3. Any reset (POR, external, software, COP, and so on).
STOP	Master clock generation in the OCCS remains operational, but the SIM disables the generation of system and peripheral clocks.		Core executes STOP instruction to enter this mode. Possible recoveries from stop mode to RUN mode are: 1. Interrupt from any peripheral configured in SD register to operate in STOP mode. 2. Low voltage interrupt 3. Executing a debug mode entry command using the core JTAG interface. 4. Any reset.

RUN, WAIT and STOP modes provide means of enabling/disabling the peripheral and/or system clocks as a group. Peripherals must be enabled (refer to PCEn registers) to operate in any mode. Once enabled, their standard behavior is to operate in RUN and WAIT modes but to be disabled in STOP mode. However, by asserting a peripheral's STOP disable bit (refer to SDn registers), the peripheral clock continues to operate in STOP mode. This permits selected peripherals to remain operational in STOP mode to produce interrupts which can recover the part from STOP mode back into RUN mode.

System clocks are individually gated in the SIM based on their specific requirements as a function of low-power mode. Clocks specific to the core processor operate only in RUN mode. The DMAEbl field in the SIM Control register determines which low power modes in which clocking to the DMA is enabled. Clocks to system bus slaves including the IPBus interface, the RAM, and the flash memory, will operate in any low power mode in which either the core processor or the DMA are enabled.

The SCIs can be configured to operate at two times the system bus rate using the SCIn_CR control bits.

RUN, WAIT and STOP modes may be combined with the power management features of the PMC, flash memory low power mode feature, and clock generation configuration of the OCCS to provide a broad palette of power control techniques.

11.3.3 STOP and WAIT Mode Disable Function

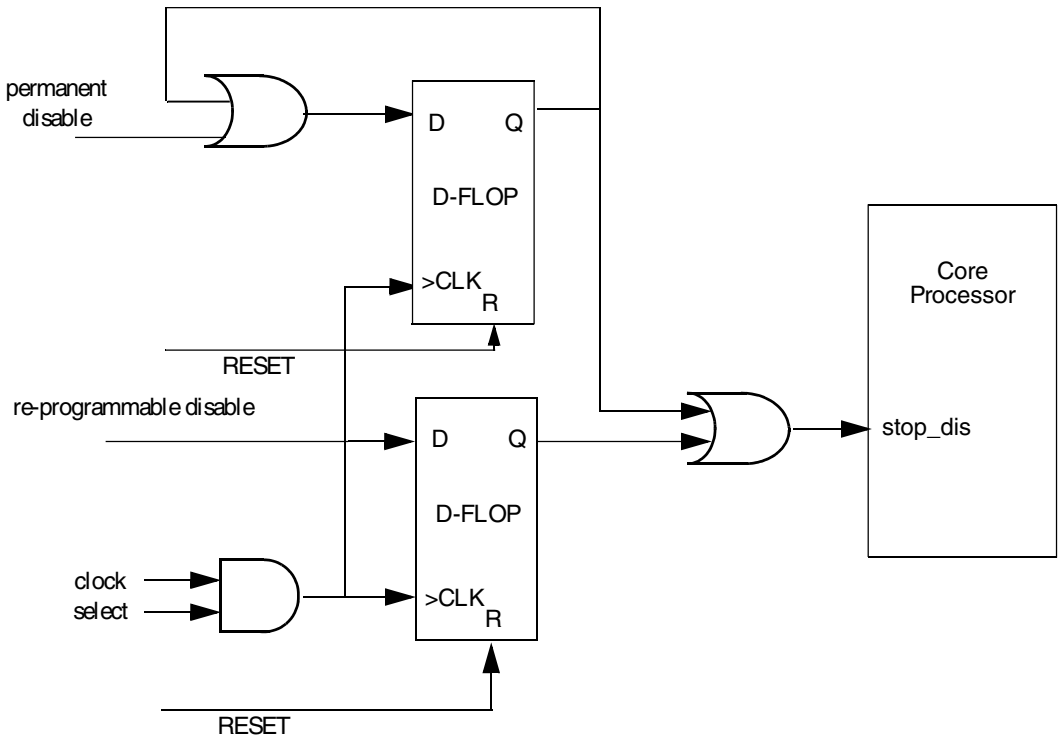


Figure 11-2. STOP Disable Circuit

The core processor supports both STOP and WAIT instructions. Both put the CPU to sleep. The peripheral bus continues to run in WAIT mode, but in STOP mode, only peripherals whose SDn control is asserted continue to run. Entry into STOP or WAIT mode does not affect the OCCS configuration and affects only the generation of system and peripheral clocks using the master clocks from the OCCS. The OCCS may be reconfigured prior to entering STOP or WAIT, if desired, to reduce master clock frequencies and thus the power utilization within the OCCS.

Some applications require the core's STOP/WAIT instructions to be disabled. Control fields are provided in the CTRL register to disable WAIT and/or STOP modes. This setting can be made irrecoverable (recoverable only at the next reset) as illustrated in Figure 11-2, by setting the lock bit within these fields.

11.4 Resets

The SIM supports several sources of reset.

Table 11-2. Sources of Reset

Label	Source of Reset	Timing
EXTR	External Reset	Asynchronous
POR	Power-On-Reset (PMC)	Asynchronous
COP_CPU	COP CPU reset	Asynchronous
COP_LOR	COP Loss of Reference reset	Synchronous
SWR	Software reset (SIM)	Synchronous

11.5 Clocks

All system and peripheral clocks are derived in the SIM by dividing the MSTR_2X clock from OCCS by two. Exceptions include the RAM system clock, which is a gated version of the MSTR_2X clock, and the 2x clock option to the PWMA and SCI modules. These clocks operate at up to the MSTR_2X clock's maximum frequency. The SIM is responsible for stalling individual clocks as a response to holdoff requests from system bus slaves, low power modes, and other clock configuration parameters.

11.6 Interrupts

The SIM generates no interrupts.

Chapter 12
Interrupt Controller (INTC)

12.1 Introduction

The Interrupt Controller (INTC) module arbitrates among the various interrupt requests (IRQs). The module supports unique interrupt vectors and programmable interrupt priority. It signals to the core when an interrupt of sufficient priority exists and to what address to jump to service this interrupt.

12.1.1 References

- DSP56800E and DSP56800EX Core Reference Manual

12.1.2 Features

The INTC module design has these distinctive features:

- Programmable priority levels for each IRQ
- Two programmable fast interrupts
- Notification to System Integration Module (SIM) to restart clocks when exiting wait and stop modes
- Driving of initial address on the address bus after reset

12.1.3 Modes of Operation

The INTC module design has these major modes of operation: functional mode and wait and stop modes.

- Functional mode

  The INTC is in this mode by default.

- Wait and stop mode operation

  In wait and stop modes, the system clocks and the core are turned off. The INTC signals a pending IRQ to the System Integration Module (SIM) to restart the clocks and service the IRQ. An IRQ can wake up the core only if the IRQ is enabled prior to entering wait or stop mode.

12.1.4 Block Diagram

The following figure is a block diagram of the INTC module.

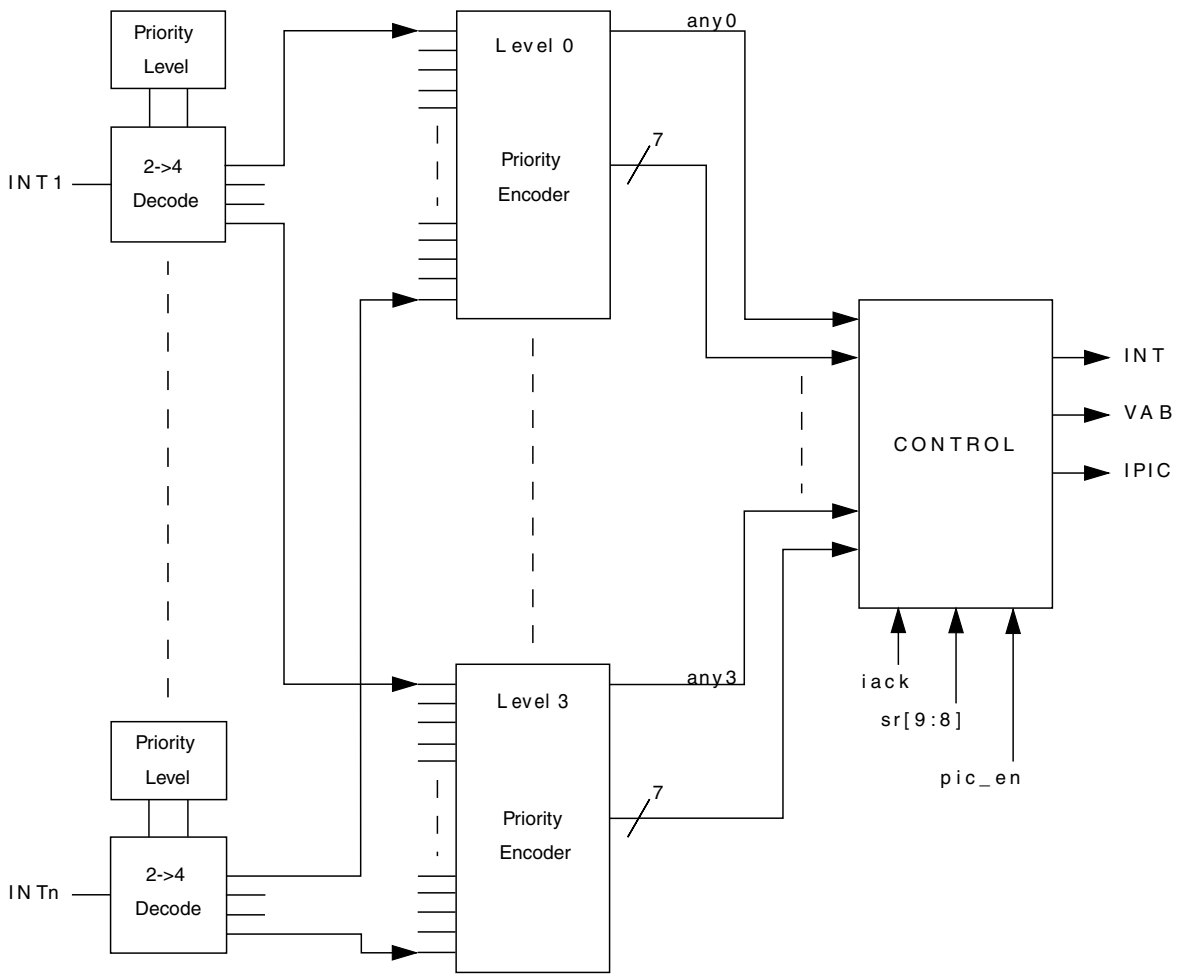


Figure 12-1. Interrupt Controller Block Diagram

12.2 Memory Map and Registers

This module uses 16-bit word addressing.

INTC memory map

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
E300	Interrupt Priority Register 0 (INTC_IPR0)	16	R/W	0000h	12.2.1/222
E301	Interrupt Priority Register 1 (INTC_IPR1)	16	R/W	0000h	12.2.2/223
E302	Interrupt Priority Register 2 (INTC_IPR2)	16	R/W	0000h	12.2.3/225
E303	Interrupt Priority Register 3 (INTC_IPR3)	16	R/W	0000h	12.2.4/226
E304	Interrupt Priority Register 4 (INTC_IPR4)	16	R/W	0000h	12.2.5/228
E305	Interrupt Priority Register 5 (INTC_IPR5)	16	R/W	0000h	12.2.6/229
E306	Interrupt Priority Register 6 (INTC_IPR6)	16	R/W	0000h	12.2.7/231
E308	Interrupt Priority Register 8 (INTC_IPR8)	16	R/W	0000h	12.2.8/232
E309	Interrupt Priority Register 9 (INTC_IPR9)	16	R/W	0000h	12.2.9/233
E30A	Interrupt Priority Register 10 (INTC_IPR10)	16	R/W	0000h	12.2.10/234
E30B	Interrupt Priority Register 11 (INTC_IPR11)	16	R/W	0000h	12.2.11/236
E30C	Interrupt Priority Register 12 (INTC_IPR12)	16	R/W	0000h	12.2.12/237
E30D	Vector Base Address Register (INTC_VBA)	16	R/W	0000h	12.2.13/238
E30E	Fast Interrupt 0 Match Register (INTC_FIM0)	16	R/W	0000h	12.2.14/239
E30F	Fast Interrupt 0 Vector Address Low Register (INTC_FIVAL0)	16	R/W	0000h	12.2.15/239
E310	Fast Interrupt 0 Vector Address High Register (INTC_FIVAH0)	16	R/W	0000h	12.2.16/240
E311	Fast Interrupt 1 Match Register (INTC_FIM1)	16	R/W	0000h	12.2.17/240
E312	Fast Interrupt 1 Vector Address Low Register (INTC_FIVAL1)	16	R/W	0000h	12.2.18/241
E313	Fast Interrupt 1 Vector Address High Register (INTC_FIVAH1)	16	R/W	0000h	12.2.19/241
E314	IRQ Pending Register 0 (INTC_IRQP0)	16	R	FFFFh	12.2.20/241
E315	IRQ Pending Register 1 (INTC_IRQP1)	16	R	FFFFh	12.2.21/242
E316	IRQ Pending Register 2 (INTC_IRQP2)	16	R	FFFFh	12.2.22/242
E317	IRQ Pending Register 3 (INTC_IRQP3)	16	R	FFFFh	12.2.23/243
E318	IRQ Pending Register 4 (INTC_IRQP4)	16	R	FFFFh	12.2.24/243
E319	IRQ Pending Register 5 (INTC_IRQP5)	16	R	FFFFh	12.2.25/244
E31A	IRQ Pending Register 6 (INTC_IRQP6)	16	R	FFFFh	12.2.26/244
E31B	Control Register (INTC_CTRL)	16	R/W	001Ch	12.2.27/245

12.2.1 Interrupt Priority Register 0 (INTC_IPR0)

Address: E300h base + 0h offset = E300h

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read Write	Reserved		Reserved		BUS_ERR		RX_REG		TX_REG		TRBUF		BKPT		STPCNT	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

INTC_IPR0 field descriptions

Field	Description
15–14 Reserved	This field is reserved. Do not modify this bitfield.
13–12 Reserved	This field is reserved. Do not modify this bitfield.
11–10 BUS_ERR	Bus Error Interrupt Priority Level These fields are used to set the interrupt priority levels for certain IRQs. These IRQs are limited to priorities 1-3 and are disabled by default. 00 IRQ disabled (default) 01 IRQ Priority Level 1 10 IRQ Priority Level 2 11 IRQ Priority Level 3
9–8 RX_REG	EOnCE Receive Register Full Interrupt Priority Level These fields are used to set the interrupt priority levels for certain IRQs. These IRQs are limited to priorities 1-3 and are disabled by default. 00 IRQ disabled (default) 01 IRQ Priority Level 1 10 IRQ Priority Level 2 11 IRQ Priority Level 3
7–6 TX_REG	EOnCE Transmit Register Empty Interrupt Priority Level These fields are used to set the interrupt priority levels for certain IRQs. These IRQs are limited to priorities 1-3 and are disabled by default. 00 IRQ disabled (default) 01 IRQ Priority Level 1 10 IRQ Priority Level 2 11 IRQ Priority Level 3
5–4 TRBUF	EOnCE Trace Buffer Interrupt Priority Level These fields are used to set the interrupt priority levels for certain IRQs. These IRQs are limited to priorities 1-3 and are disabled by default. 00 IRQ disabled (default) 01 IRQ Priority Level 1 10 IRQ Priority Level 2 11 IRQ Priority Level 3

Table continues on the next page...

INTC_IPR0 field descriptions (continued)

Field	Description
3–2 BKPT	EOnCE Breakpoint Interrupt Priority Level These fields are used to set the interrupt priority levels for certain IRQs. These IRQs are limited to priorities 1-3 and are disabled by default. 00 IRQ disabled (default) 01 IRQ Priority Level 1 10 IRQ Priority Level 2 11 IRQ Priority Level 3
STPCNT	EOnCE Step Counter Interrupt Priority Level These fields are used to set the interrupt priority levels for certain IRQs. These IRQs are limited to priorities 1-3 and are disabled by default. 00 IRQ disabled (default) 01 IRQ Priority Level 1 10 IRQ Priority Level 2 11 IRQ Priority Level 3

12.2.2 Interrupt Priority Register 1 (INTC_IPR1)

Address: E300h base + 1h offset = E301h

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read Write	TMRB_0		TMRB_1		TMRB_2		TMRB_3		OCCS		LVI1		XBARA		0	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

INTC_IPR1 field descriptions

Field	Description
15–14 TMRB_0	Timer B Channel 0 Interrupt Priority Level These fields are used to set the interrupt priority levels for certain peripheral IRQs. These IRQs are limited to priorities 0-2 and are disabled by default. 00 IRQ disabled (default) 01 IRQ Priority Level 0 10 IRQ Priority Level 1 11 IRQ Priority Level 2
13–12 TMRB_1	Timer B Channel 1 Interrupt Priority Level These fields are used to set the interrupt priority levels for certain peripheral IRQs. These IRQs are limited to priorities 0-2 and are disabled by default. 00 IRQ disabled (default) 01 IRQ Priority Level 0 10 IRQ Priority Level 1 11 IRQ Priority Level 2

Table continues on the next page...

INTC_IPR1 field descriptions (continued)

Field	Description
11–10 TMRB_2	Timer B Channel 2 Interrupt Priority Level These fields are used to set the interrupt priority levels for certain peripheral IRQs. These IRQs are limited to priorities 0-2 and are disabled by default. 00 IRQ disabled (default) 01 IRQ Priority Level 0 10 IRQ Priority Level 1 11 IRQ Priority Level 2
9–8 TMRB_3	Timer B Channel 3 Interrupt Priority Level This field is used to set the interrupt priority levels for certain peripheral IRQs. These IRQs are limited to priorities 0-2 and are disabled by default. 00 IRQ disabled (default) 01 IRQ Priority Level 0 10 IRQ Priority Level 1 11 IRQ Priority Level 2
7–6 OCCS	PLL Loss of Reference or Change in Lock Status Interrupt Priority Level These fields are used to set the interrupt priority levels for certain IRQs. These IRQs are limited to priorities 1-3 and are disabled by default. 00 IRQ disabled (default) 01 IRQ Priority Level 1 10 IRQ Priority Level 2 11 IRQ Priority Level 3
5–4 LVI1	Low Voltage Detector Interrupt Priority Level These fields are used to set the interrupt priority levels for certain IRQs. These IRQs are limited to priorities 1-3 and are disabled by default. 00 IRQ disabled (default) 01 IRQ Priority Level 1 10 IRQ Priority Level 2 11 IRQ Priority Level 3
3–2 XBARA	Inter-Peripheral Crossbar Switch A (XBARA) Interrupt Priority Level These fields are used to set the interrupt priority levels for certain IRQs. These IRQs are limited to priorities 1-3 and are disabled by default. 00 IRQ disabled (default) 01 IRQ Priority Level 1 10 IRQ Priority Level 2 11 IRQ Priority Level 3
Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

12.2.3 Interrupt Priority Register 2 (INTC_IPR2)

Address: E300h base + 2h offset = E302h

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read Write	ADC_COCO		ADC_ERR		ADC_CC0		ADC_CC1		TMRA_0		TMRA_1		TMRA_2		TMRA_3	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

INTC_IPR2 field descriptions

Field	Description
15–14 ADC_COCO	ADC_SAR Conversion Complete Interrupt Priority Level These fields are used to set the interrupt priority levels for certain peripheral IRQs. These IRQs are limited to priorities 0-2 and are disabled by default. 00 IRQ disabled (default) 01 IRQ Priority Level 0 10 IRQ Priority Level 1 11 IRQ Priority Level 2
13–12 ADC_ERR	ADC_CYC Zero Crossing, High Limit, or Low Limit Interrupt Priority Level These fields are used to set the interrupt priority levels for certain peripheral IRQs. These IRQs are limited to priorities 0-2 and are disabled by default. 00 IRQ disabled (default) 01 IRQ Priority Level 0 10 IRQ Priority Level 1 11 IRQ Priority Level 2
11–10 ADC_CC0	ADC_CYC Conversion Complete Interrupt Priority Level (any scan type except converter B in non-simultaneous parallel scan mode) These fields are used to set the interrupt priority levels for certain peripheral IRQs. These IRQs are limited to priorities 0-2 and are disabled by default. 00 IRQ disabled (default) 01 IRQ Priority Level 0 10 IRQ Priority Level 1 11 IRQ Priority Level 2
9–8 ADC_CC1	ADC_CYC Conversion Complete Interrupt Priority Level (converter B in non-simultaneous parallel scan mode) These fields are used to set the interrupt priority levels for certain peripheral IRQs. These IRQs are limited to priorities 0-2 and are disabled by default. 00 IRQ disabled (default) 01 IRQ Priority Level 0 10 IRQ Priority Level 1 11 IRQ Priority Level 2
7–6 TMRA_0	Timer A Channel 0 Interrupt Priority Level

Table continues on the next page...

INTC_IPR2 field descriptions (continued)

Field	Description
	These fields are used to set the interrupt priority levels for certain peripheral IRQs. These IRQs are limited to priorities 0-2 and are disabled by default. 00 IRQ disabled (default) 01 IRQ Priority Level 0 10 IRQ Priority Level 1 11 IRQ Priority Level 2
5–4 TMRA_1	Timer A Channel 1 Interrupt Priority Level These fields are used to set the interrupt priority levels for certain peripheral IRQs. These IRQs are limited to priorities 0-2 and are disabled by default. 00 IRQ disabled (default) 01 IRQ Priority Level 0 10 IRQ Priority Level 1 11 IRQ Priority Level 2
3–2 TMRA_2	Timer A Channel 2 Interrupt Priority Level These fields are used to set the interrupt priority levels for certain peripheral IRQs. These IRQs are limited to priorities 0-2 and are disabled by default. 00 IRQ disabled (default) 01 IRQ Priority Level 0 10 IRQ Priority Level 1 11 IRQ Priority Level 2
TMRA_3	Timer A Channel 3 Interrupt Priority Level These fields are used to set the interrupt priority levels for certain peripheral IRQs. These IRQs are limited to priorities 0-2 and are disabled by default. 00 IRQ disabled (default) 01 IRQ Priority Level 0 10 IRQ Priority Level 1 11 IRQ Priority Level 2

12.2.4 Interrupt Priority Register 3 (INTC_IPR3)

Address: E300h base + 3h offset = E303h

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read Write	CAN_TX_WARN		CAN_ERROR		CAN_BUS_OFF		CAN_MB_OR		DMACH0		DMACH1		DMACH2		DMACH3	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

INTC_IPR3 field descriptions

Field	Description
15–14 CAN_TX_WARN	CAN Transmit Warning Interrupt Priority Level

Table continues on the next page...

INTC_IPR3 field descriptions (continued)

Field	Description
	These fields are used to set the interrupt priority levels for certain peripheral IRQs. These IRQs are limited to priorities 0-2 and are disabled by default. 00 IRQ disabled (default) 01 IRQ Priority Level 0 10 IRQ Priority Level 1 11 IRQ Priority Level 2
13–12 CAN_ERROR	CAN Error Interrupt Priority Level These fields are used to set the interrupt priority levels for certain peripheral IRQs. These IRQs are limited to priorities 0-2 and are disabled by default. 00 IRQ disabled (default) 01 IRQ Priority Level 0 10 IRQ Priority Level 1 11 IRQ Priority Level 2
11–10 CAN_BUS_OFF	CAN Bus Off Interrupt Priority Level These fields are used to set the interrupt priority levels for certain peripheral IRQs. These IRQs are limited to priorities 0-2 and are disabled by default. 00 IRQ disabled (default) 01 IRQ Priority Level 0 10 IRQ Priority Level 1 11 IRQ Priority Level 2
9–8 CAN_MB_OR	CAN Message Buffer Interrupt Priority Level These fields are used to set the interrupt priority levels for certain peripheral IRQs. These IRQs are limited to priorities 0-2 and are disabled by default. 00 IRQ disabled (default) 01 IRQ Priority Level 0 10 IRQ Priority Level 1 11 IRQ Priority Level 2
7–6 DMACH0	DMA Channel 0 Service Request Interrupt Priority Level These fields are used to set the interrupt priority levels for certain peripheral IRQs. These IRQs are limited to priorities 0-2 and are disabled by default. 00 IRQ disabled (default) 01 IRQ Priority Level 0 10 IRQ Priority Level 1 11 IRQ Priority Level 2
5–4 DMACH1	DMA Channel 1 Service Request Interrupt Priority Level These fields are used to set the interrupt priority levels for certain peripheral IRQs. These IRQs are limited to priorities 0-2 and are disabled by default. 00 IRQ disabled (default) 01 IRQ Priority Level 0 10 IRQ Priority Level 1 11 IRQ Priority Level 2

Table continues on the next page...

INTC_IPR3 field descriptions (continued)

Field	Description
3–2 DMACH2	DMA Channel 2 Service Request Interrupt Priority Level These fields are used to set the interrupt priority levels for certain peripheral IRQs. These IRQs are limited to priorities 0-2 and are disabled by default. 00 IRQ disabled (default) 01 IRQ Priority Level 0 10 IRQ Priority Level 1 11 IRQ Priority Level 2
DMACH3	DMA Channel 3 Service Request Interrupt Priority Level These fields are used to set the interrupt priority levels for certain peripheral IRQs. These IRQs are limited to priorities 0-2 and are disabled by default. 00 IRQ disabled (default) 01 IRQ Priority Level 0 10 IRQ Priority Level 1 11 IRQ Priority Level 2

12.2.5 Interrupt Priority Register 4 (INTC_IPR4)

Address: E300h base + 4h offset = E304h

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	QSCI1_RCV		QSCI1_RERR		0								CAN_WAKEUP		CAN_RX_WARN	
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

INTC_IPR4 field descriptions

Field	Description
15–14 QSCI1_RCV	QSCI 1 Receive Data Register Full Interrupt Priority Level These fields are used to set the interrupt priority levels for certain peripheral IRQs. These IRQs are limited to priorities 0-2 and are disabled by default. 00 IRQ disabled (default) 01 IRQ Priority Level 0 10 IRQ Priority Level 1 11 IRQ Priority Level 2
13–12 QSCI1_RERR	QSCI 1 Receiver Error Interrupt Priority Level These fields are used to set the interrupt priority levels for certain peripheral IRQs. These IRQs are limited to priorities 0-2 and are disabled by default. 00 IRQ disabled (default) 01 IRQ Priority Level 0 10 IRQ Priority Level 1 11 IRQ Priority Level 2

Table continues on the next page...

INTC_IPR4 field descriptions (continued)

Field	Description
11–4 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
3–2 CAN_WAKEUP	CAN Wakeup Interrupt Priority Level These fields are used to set the interrupt priority levels for certain peripheral IRQs. These IRQs are limited to priorities 0-2 and are disabled by default. 00 IRQ disabled (default) 01 IRQ Priority Level 0 10 IRQ Priority Level 1 11 IRQ Priority Level 2
CAN_RX_WARN	CAN Receive Warning Interrupt Priority Level These fields are used to set the interrupt priority levels for certain peripheral IRQs. These IRQs are limited to priorities 0-2 and are disabled by default. 00 IRQ disabled (default) 01 IRQ Priority Level 0 10 IRQ Priority Level 1 11 IRQ Priority Level 2

12.2.6 Interrupt Priority Register 5 (INTC_IPR5)

Address: E300h base + 5h offset = E305h

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	0		0		QSCI0_ TDRE		QSCI0_ TIDLE		QSCI0_RCV		QSCI0_ RERR		QSCI1_ TDRE		QSCI1_ TIDLE	
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

INTC_IPR5 field descriptions

Field	Description
15–14 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
13–12 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
11–10 QSCI0_TDRE	QSCI 0 Transmit Data Register Empty Interrupt Priority Level These fields are used to set the interrupt priority levels for certain peripheral IRQs. These IRQs are limited to priorities 0-2 and are disabled by default. 00 IRQ disabled (default) 01 IRQ Priority Level 0 10 IRQ Priority Level 1 11 IRQ Priority Level 2
9–8 QSCI0_TIDLE	QSCI 0 Transmitter Idle Interrupt Priority Level

Table continues on the next page...

INTC_IPR5 field descriptions (continued)

Field	Description
	These fields are used to set the interrupt priority levels for certain peripheral IRQs. These IRQs are limited to priorities 0-2 and are disabled by default. 00 IRQ disabled (default) 01 IRQ Priority Level 0 10 IRQ Priority Level 1 11 IRQ Priority Level 2
7–6 QSCI0_RCV	QSCI 0 Receive Data Register Full Interrupt Priority Level These fields are used to set the interrupt priority levels for certain peripheral IRQs. These IRQs are limited to priorities 0-2 and are disabled by default. 00 IRQ disabled (default) 01 IRQ Priority Level 0 10 IRQ Priority Level 1 11 IRQ Priority Level 2
5–4 QSCI0_RERR	QSCI0 Receiver Error Interrupt Priority Level These fields are used to set the interrupt priority levels for certain peripheral IRQs. These IRQs are limited to priorities 0-2 and are disabled by default. 00 IRQ disabled (default) 01 IRQ Priority Level 0 10 IRQ Priority Level 1 11 IRQ Priority Level 2
3–2 QSCI1_TDRE	QSCI 1 Transmit Data Register Empty Interrupt Priority Level These fields are used to set the interrupt priority levels for certain peripheral IRQs. These IRQs are limited to priorities 0-2 and are disabled by default. 00 IRQ disabled (default) 01 IRQ Priority Level 0 10 IRQ Priority Level 1 11 IRQ Priority Level 2
QSCI1_TIDLE	QSCI 1 Transmitter Idle Interrupt Priority Level These fields are used to set the interrupt priority levels for certain peripheral IRQs. These IRQs are limited to priorities 0-2 and are disabled by default. 00 IRQ disabled (default) 01 IRQ Priority Level 0 10 IRQ Priority Level 1 11 IRQ Priority Level 2

12.2.7 Interrupt Priority Register 6 (INTC_IPR6)

Address: E300h base + 6h offset = E306h

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	0				IIC0		IIC1		QSPI0_RCV		QSPI0_XMIT		0			
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

INTC_IPR6 field descriptions

Field	Description
15–12 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
11–10 IIC0	IIC0 Interrupt Priority Level These fields are used to set the interrupt priority levels for certain peripheral IRQs. These IRQs are limited to priorities 0-2 and are disabled by default. 00 IRQ disabled (default) 01 IRQ Priority Level 0 10 IRQ Priority Level 1 11 IRQ Priority Level 2
9–8 IIC1	IIC1 Interrupt Priority Level These fields are used to set the interrupt priority levels for certain peripheral IRQs. These IRQs are limited to priorities 0-2 and are disabled by default. 00 IRQ disabled (default) 01 IRQ Priority Level 0 10 IRQ Priority Level 1 11 IRQ Priority Level 2
7–6 QSPI0_RCV	QSPI0 Receiver Full Interrupt Priority Level These fields are used to set the interrupt priority levels for certain peripheral IRQs. These IRQs are limited to priorities 0-2 and are disabled by default. 00 IRQ disabled (default) 01 IRQ Priority Level 0 10 IRQ Priority Level 1 11 IRQ Priority Level 2
5–4 QSPI0_XMIT	QSPI0 Transmitter Empty Interrupt Priority Level These fields are used to set the interrupt priority levels for certain peripheral IRQs. These IRQs are limited to priorities 0-2 and are disabled by default. 00 IRQ disabled (default) 01 IRQ Priority Level 0 10 IRQ Priority Level 1 11 IRQ Priority Level 2
Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

12.2.8 Interrupt Priority Register 8 (INTC_IPR8)

Address: E300h base + 8h offset = E308h

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	PWMA_CAP		PWMA_ RELOAD3		PWMA_ RERR		PWMA_ FAULT		0							
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

INTC_IPR8 field descriptions

Field	Description
15–14 PWMA_CAP	PWMA Submodule Capture Interrupt Priority Level These fields are used to set the interrupt priority levels for certain peripheral IRQs. These IRQs are limited to priorities 0-2 and are disabled by default. The IRQ represented by this field is a logical OR of input captures for PWMA's submodules 0-3: PWMA_CAP3 | PWMA_CAP2 | PWMA_CAP1 | PWMA_CAP0. 00 IRQ disabled (default) 01 IRQ Priority Level 0 10 IRQ Priority Level 1 11 IRQ Priority Level 2
13–12 PWMA_ RELOAD3	PWMA Submodule 3 Reload Interrupt Priority Level These fields are used to set the interrupt priority levels for certain peripheral IRQs. These IRQs are limited to priorities 0-2 and are disabled by default. 00 IRQ disabled (default) 01 IRQ Priority Level 0 10 IRQ Priority Level 1 11 IRQ Priority Level 2
11–10 PWMA_RERR	PWMA Reload Error Interrupt Priority Level These fields are used to set the interrupt priority levels for certain peripheral IRQs. These IRQs are limited to priorities 0-2 and are disabled by default. 00 IRQ disabled (default) 01 IRQ Priority Level 0 10 IRQ Priority Level 1 11 IRQ Priority Level 2
9–8 PWMA_FAULT	PWMA Fault Interrupt Priority Level These fields are used to set the interrupt priority levels for certain peripheral IRQs. These IRQs are limited to priorities 0-2 and are disabled by default. 00 IRQ disabled (default) 01 IRQ Priority Level 0 10 IRQ Priority Level 1 11 IRQ Priority Level 2
Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

12.2.9 Interrupt Priority Register 9 (INTC_IPR9)

Address: E300h base + 9h offset = E309h

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read Write	FTFL_RDCOL		PWMA_CMP0		PWMA_RELOAD0		PWMA_CMP1		PWMA_RELOAD1		PWMA_CMP2		PWMA_RELOAD2		PWMA_CMP3	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

INTC_IPR9 field descriptions

Field	Description
15–14 FTFL_RDCOL	FTFL Access Error Priority Level These fields are used to set the interrupt priority levels for certain peripheral IRQs. These IRQs are limited to priorities 0-2 and are disabled by default. 00 IRQ disabled (default) 01 IRQ Priority Level 0 10 IRQ Priority Level 1 11 IRQ Priority Level 2
13–12 PWMA_CMP0	PWMA Submodule 0 Compare Interrupt Priority Level These fields are used to set the interrupt priority levels for certain peripheral IRQs. These IRQs are limited to priorities 0-2 and are disabled by default. 00 IRQ disabled (default) 01 IRQ Priority Level 0 10 IRQ Priority Level 1 11 IRQ Priority Level 2
11–10 PWMA_ RELOAD0	PWMA Submodule 0 Reload Interrupt Priority Level These fields are used to set the interrupt priority levels for certain peripheral IRQs. These IRQs are limited to priorities 0-2 and are disabled by default. 00 IRQ disabled (default) 01 IRQ Priority Level 0 10 IRQ Priority Level 1 11 IRQ Priority Level 2
9–8 PWMA_CMP1	PWMA Submodule 1 Compare Interrupt Priority Level These fields are used to set the interrupt priority levels for certain peripheral IRQs. These IRQs are limited to priorities 0-2 and are disabled by default. 00 IRQ disabled (default) 01 IRQ Priority Level 0 10 IRQ Priority Level 1 11 IRQ Priority Level 2
7–6 PWMA_ RELOAD1	PWMA Submodule 1 Reload Interrupt Priority Level These fields are used to set the interrupt priority levels for certain peripheral IRQs. These IRQs are limited to priorities 0-2 and are disabled by default.

Table continues on the next page...

INTC_IPR9 field descriptions (continued)

Field	Description
	00 IRQ disabled (default) 01 IRQ Priority Level 0 10 IRQ Priority Level 1 11 IRQ Priority Level 2
5–4 PWMA_CMP2	PWMA Submodule 2 Compare Interrupt Priority Level These fields are used to set the interrupt priority levels for certain peripheral IRQs. These IRQs are limited to priorities 0-2 and are disabled by default. 00 IRQ disabled (default) 01 IRQ Priority Level 0 10 IRQ Priority Level 1 11 IRQ Priority Level 2
3–2 PWMA_ RELOAD2	PWMA Submodule 2 Reload Interrupt Priority Level These fields are used to set the interrupt priority levels for certain peripheral IRQs. These IRQs are limited to priorities 0-2 and are disabled by default. 00 IRQ disabled (default) 01 IRQ Priority Level 0 10 IRQ Priority Level 1 11 IRQ Priority Level 2
PWMA_CMP3	PWMA Submodule 3 Compare Interrupt Priority Level These fields are used to set the interrupt priority levels for certain peripheral IRQs. These IRQs are limited to priorities 0-2 and are disabled by default. 00 IRQ disabled (default) 01 IRQ Priority Level 0 10 IRQ Priority Level 1 11 IRQ Priority Level 2

12.2.10 Interrupt Priority Register 10 (INTC_IPR10)

Address: E300h base + Ah offset = E30Ah

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read Write	PDB1		PIT0_ ROLLOVR		PIT1_ ROLLOVR		CMPA		CMPB		CMPC		CMPD		FTFL_CC	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

INTC_IPR10 field descriptions

Field	Description
15–14 PDB1	PDB1 Interrupt Priority Level These fields are used to set the interrupt priority levels for certain peripheral IRQs. These IRQs are limited to priorities 0-2 and are disabled by default. 00 IRQ disabled (default)

Table continues on the next page...

INTC_IPR10 field descriptions (continued)

Field	Description
	01 IRQ Priority Level 0 10 IRQ Priority Level 1 11 IRQ Priority Level 2
13–12 PIT0_ROLLOVR	PIT0 Roll Over Interrupt Priority Level These fields are used to set the interrupt priority levels for certain peripheral IRQs. These IRQs are limited to priorities 0-2 and are disabled by default. 00 IRQ disabled (default) 01 IRQ Priority Level 0 10 IRQ Priority Level 1 11 IRQ Priority Level 2
11–10 PIT1_ROLLOVR	PIT1 Roll Over Interrupt Priority Level These fields are used to set the interrupt priority levels for certain peripheral IRQs. These IRQs are limited to priorities 0-2 and are disabled by default. 00 IRQ disabled (default) 01 IRQ Priority Level 0 10 IRQ Priority Level 1 11 IRQ Priority Level 2
9–8 CMPA	Comparator A Interrupt Priority Level These fields are used to set the interrupt priority levels for certain peripheral IRQs. These IRQs are limited to priorities 0-2 and are disabled by default. 00 IRQ disabled (default) 01 IRQ Priority Level 0 10 IRQ Priority Level 1 11 IRQ Priority Level 2
7–6 CMPB	Comparator B Interrupt Priority Level These fields are used to set the interrupt priority levels for certain peripheral IRQs. These IRQs are limited to priorities 0-2 and are disabled by default. 00 IRQ disabled (default) 01 IRQ Priority Level 0 10 IRQ Priority Level 1 11 IRQ Priority Level 2
5–4 CMPC	Comparator C Interrupt Priority Level These fields are used to set the interrupt priority levels for certain peripheral IRQs. These IRQs are limited to priorities 0-2 and are disabled by default. 00 IRQ disabled (default) 01 IRQ Priority Level 0 10 IRQ Priority Level 1 11 IRQ Priority Level 2
3–2 CMPD	Comparator D Interrupt Priority Level These fields are used to set the interrupt priority levels for certain peripheral IRQs. These IRQs are limited to priorities 0-2 and are disabled by default.

Table continues on the next page...

INTC_IPR10 field descriptions (continued)

Field	Description
	00 IRQ disabled (default) 01 IRQ Priority Level 0 10 IRQ Priority Level 1 11 IRQ Priority Level 2
FTFL_CC	FTFL Command Complete Interrupt Priority Level These fields are used to set the interrupt priority levels for certain peripheral IRQs. These IRQs are limited to priorities 0-2 and are disabled by default. 00 IRQ disabled (default) 01 IRQ Priority Level 0 10 IRQ Priority Level 1 11 IRQ Priority Level 2

12.2.11 Interrupt Priority Register 11 (INTC_IPR11)

Address: E300h base + Bh offset = E30Bh

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	GPIOD		GPIOE		GPIOF		0		0						PDB0	
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

INTC_IPR11 field descriptions

Field	Description
15–14 GPIOD	GPIO D Interrupt Priority Level These fields are used to set the interrupt priority levels for certain peripheral IRQs. These IRQs are limited to priorities 0-2 and are disabled by default. 00 IRQ disabled (default) 01 IRQ Priority Level 0 10 IRQ Priority Level 1 11 IRQ Priority Level 2
13–12 GPIOE	GPIO E Interrupt Priority Level These fields are used to set the interrupt priority levels for certain peripheral IRQs. These IRQs are limited to priorities 0-2 and are disabled by default. 00 IRQ disabled (default) 01 IRQ Priority Level 0 10 IRQ Priority Level 1 11 IRQ Priority Level 2
11–10 GPIOF	GPIO F Interrupt Priority Level These fields are used to set the interrupt priority levels for certain peripheral IRQs. These IRQs are limited to priorities 0-2 and are disabled by default.

Table continues on the next page...

INTC_IPR11 field descriptions (continued)

Field	Description
	00 IRQ disabled (default) 01 IRQ Priority Level 0 10 IRQ Priority Level 1 11 IRQ Priority Level 2
9–8 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
7–2 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
PDB0	PDB0 Interrupt Priority Level These fields are used to set the interrupt priority levels for certain peripheral IRQs. These IRQs are limited to priorities 0-2 and are disabled by default. 00 IRQ disabled (default) 01 IRQ Priority Level 0 10 IRQ Priority Level 1 11 IRQ Priority Level 2

12.2.12 Interrupt Priority Register 12 (INTC_IPR12)

Address: E300h base + Ch offset = E30Ch

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	0		0		0		EWM_INT		COP_INT		GPIOA		GPIOB		GPIOC	
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

INTC_IPR12 field descriptions

Field	Description
15–14 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
13–12 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
11–10 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
9–8 EWM_INT	External Watchdog Monitor Interrupt Priority Level These fields are used to set the interrupt priority levels for certain peripheral IRQs. These IRQs are limited to priorities 0-2 and are disabled by default. 00 IRQ disabled (default) 01 IRQ Priority Level 0 10 IRQ Priority Level 1 11 IRQ Priority Level 2
7–6 COP_INT	COP Watchdog Interrupt Priority Level

Table continues on the next page...

INTC_IPR12 field descriptions (continued)

Field	Description
	These fields are used to set the interrupt priority levels for certain peripheral IRQs. These IRQs are limited to priorities 0-2 and are disabled by default. 00 IRQ disabled (default) 01 IRQ Priority Level 0 10 IRQ Priority Level 1 11 IRQ Priority Level 2
5–4 GPIOA	GPIO A Interrupt Priority Level These fields are used to set the interrupt priority levels for certain peripheral IRQs. These IRQs are limited to priorities 0-2 and are disabled by default. 00 IRQ disabled (default) 01 IRQ Priority Level 0 10 IRQ Priority Level 1 11 IRQ Priority Level 2
3–2 GPIOB	GPIO B Interrupt Priority Level These fields are used to set the interrupt priority levels for certain peripheral IRQs. These IRQs are limited to priorities 0-2 and are disabled by default. 00 IRQ disabled (default) 01 IRQ Priority Level 0 10 IRQ Priority Level 1 11 IRQ Priority Level 2
GPIOC	GPIO C Interrupt Priority Level These fields are used to set the interrupt priority levels for certain peripheral IRQs. These IRQs are limited to priorities 0-2 and are disabled by default. 00 IRQ disabled (default) 01 IRQ Priority Level 0 10 IRQ Priority Level 1 11 IRQ Priority Level 2

12.2.13 Vector Base Address Register (INTC_VBA)

Address: E300h base + Dh offset = E30Dh

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	0			VECTOR_BASE_ADDRESS												
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

INTC_VBA field descriptions

Field	Description
15–13 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

Table continues on the next page...

INTC_VBA field descriptions (continued)

Field	Description
VECTOR_BASE_ ADDRESS	Interrupt Vector Base Address The value in this register is used as the upper 13 bits of the interrupt vector VAB[20:0]. The lower 8 bits are determined based on the highest priority interrupt and are then appended onto the VECTOR_BASE_ADDRESS before presenting the full Vector Address Bus [VAB] to the Core. **NOTE:** After power-on reset, the device must boot from program flash at 0x00_0000, so VBA always resets to zeros.

12.2.14 Fast Interrupt 0 Match Register (INTC_FIM0)

Address: E300h base + Eh offset = E30Eh

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	0								FAST_INTERRUPT_0							
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

INTC_FIM0 field descriptions

Field	Description
15–7 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
FAST_ INTERRUPT_0	Fast Interrupt 0 Vector Number These values are used to declare which two IRQs will be Fast Interrupts. Fast Interrupts vector directly to a service routine based on values in the Fast Interrupt Vector Address registers without having to go to a jump table first. IRQs used as fast interrupts MUST be set to priority level 2. Unexpected results will occur if a fast interrupt vector is set to any other priority. Fast interrupts automatically become the highest priority level 2 interrupt regardless of their location in the interrupt table prior to being declared as fast interrupt. Fast interrupt 0 has priority over fast interrupt 1. To determine the vector number of each IRQ, refer to the vector table in the memory section of the system specification in this document.

12.2.15 Fast Interrupt 0 Vector Address Low Register (INTC_FIVAL0)

This low register is combined with the corresponding high register to form the 21-bit vector address for the fast interrupt that is defined in the corresponding match register.

Address: E300h base + Fh offset = E30Fh

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	FI_0_VECTOR_ADDRESS_LOW															
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

INTC_FIVAL0 field descriptions

Field	Description
FI_0_VECTOR_ ADDRESS_LOW	Lower 16 bits of vector address for fast interrupt 0

12.2.16 Fast Interrupt 0 Vector Address High Register (INTC_FIVAH0)

This high register is combined with the corresponding low register to form the 21-bit vector address for the fast interrupt that is defined in the corresponding match register.

Address: E300h base + 10h offset = E310h

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	0								FI_0_VECTOR_ADDRESS_HIGH							
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

INTC_FIVAH0 field descriptions

Field	Description
15–5 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
FI_0_VECTOR_ ADDRESS_HIGH	Upper 5 bits of vector address for fast interrupt 0

12.2.17 Fast Interrupt 1 Match Register (INTC_FIM1)

Address: E300h base + 11h offset = E311h

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	0									FAST_INTERRUPT_1						
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

INTC_FIM1 field descriptions

Field	Description
15–7 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
FAST_ INTERRUPT_1	Fast Interrupt 1 Vector Number These values are used to declare which two IRQs will be Fast Interrupts. Fast Interrupts vector directly to a service routine based on values in the Fast Interrupt Vector Address registers without having to go to a jump table first. IRQs used as fast interrupts MUST be set to priority level 2. Unexpected results will occur if a fast interrupt vector is set to any other priority. Fast interrupts automatically become the highest priority level 2 interrupt regardless of their location in the interrupt table prior to being declared as fast interrupt. Fast interrupt 0 has priority over fast interrupt 1. To determine the vector number of each IRQ, refer to the vector table in the memory section of the system specification in this document.

12.2.18 Fast Interrupt 1 Vector Address Low Register (INTC_FIVAL1)

This low register is combined with the corresponding high register to form the 21-bit vector address for the fast interrupt that is defined in the corresponding match register.

Address: E300h base + 12h offset = E312h

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read Write	FI_1_VECTOR_ADDRESS_LOW															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

INTC_FIVAL1 field descriptions

Field	Description
FI_1_VECTOR_ADDRESS_LOW	Lower 16 bits of vector address for fast interrupt 1

12.2.19 Fast Interrupt 1 Vector Address High Register (INTC_FIVAH1)

This high register is combined with the corresponding low register to form the 21-bit vector address for the fast interrupt that is defined in the corresponding match register.

Address: E300h base + 13h offset = E313h

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	0											FI_1_VECTOR_ADDRESS_HIGH				
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

INTC_FIVAH1 field descriptions

Field	Description
15–5 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
FI_1_VECTOR_ADDRESS_HIGH	Upper 5 bits of vector address for fast interrupt 1

12.2.20 IRQ Pending Register 0 (INTC_IRQP0)

Address: E300h base + 14h offset = E314h

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	PENDING[16:2]															1
Write																
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

INTC_IRQP0 field descriptions

Field	Description
15–1 PENDING[16:2]	Pending IRQs These registers combine to represent the pending IRQs for interrupt vector numbers 2 onward. **NOTE:** For interrupt vector numbers 2-16, the edge triggered IRQ's status shown by the corresponding INTC_IRQPn[PENDING] bit depends on when the IRQ Pending register is read. Before the ISR is entered, the PENDING bit shows that the IRQ is pending. After the ISR is entered, the PENDING bit shows that the IRQ is not pending. 0 IRQ pending for this vector number 1 No IRQ pending for this vector number
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 1.

12.2.21 IRQ Pending Register 1 (INTC_IRQP1)

Address: E300h base + 15h offset = E315h

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	PENDING[32:17]															
Write																
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

INTC_IRQP1 field descriptions

Field	Description
PENDING[32:17]	Pending IRQs These registers combine to represent the pending IRQs for interrupt vector numbers 2 onward. **NOTE:** For interrupt vector numbers 17-20, the edge triggered IRQ's status shown by the corresponding INTC_IRQPn[PENDING] bit depends on when the IRQ Pending register is read. Before the ISR is entered, the PENDING bit shows that the IRQ is pending. After the ISR is entered, the PENDING bit shows that the IRQ is not pending. 0 IRQ pending for this vector number 1 No IRQ pending for this vector number

12.2.22 IRQ Pending Register 2 (INTC_IRQP2)

Address: E300h base + 16h offset = E316h

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	PENDING[48:33]															
Write																
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

INTC_IRQP2 field descriptions

Field	Description
PENDING[48:33]	Pending IRQs These registers combine to represent the pending IRQs for interrupt vector numbers 2 onward. 0 IRQ pending for this vector number 1 No IRQ pending for this vector number

12.2.23 IRQ Pending Register 3 (INTC_IRQP3)

Address: E300h base + 17h offset = E317h

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	PENDING[64:49]															
Write																
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

INTC_IRQP3 field descriptions

Field	Description
PENDING[64:49]	Pending IRQs These registers combine to represent the pending IRQs for interrupt vector numbers 2 onward. 0 IRQ pending for this vector number 1 No IRQ pending for this vector number

12.2.24 IRQ Pending Register 4 (INTC_IRQP4)

Address: E300h base + 18h offset = E318h

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	PENDING[80:65]															
Write																
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

INTC_IRQP4 field descriptions

Field	Description
PENDING[80:65]	Pending IRQs These registers combine to represent the pending IRQs for interrupt vector numbers 2 onward. 0 IRQ pending for this vector number 1 No IRQ pending for this vector number

12.2.25 IRQ Pending Register 5 (INTC_IRQP5)

Address: E300h base + 19h offset = E319h

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	PENDING[96:81]															
Write																
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

INTC_IRQP5 field descriptions

Field	Description
PENDING[96:81]	Pending IRQs These registers combine to represent the pending IRQs for interrupt vector numbers 2 onward. 0 IRQ pending for this vector number 1 No IRQ pending for this vector number

12.2.26 IRQ Pending Register 6 (INTC_IRQP6)

Address: E300h base + 1Ah offset = E31Ah

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	1		PENDING[110:97]													
Write																
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

INTC_IRQP6 field descriptions

Field	Description
15–14 Reserved	This field is reserved. This read-only field is reserved and always has the value 1.
PENDING[110:97]	Pending IRQs These registers combine to represent the pending IRQs for interrupt vector numbers 2 onward. **NOTE:** For interrupt vector number 110, the edge triggered IRQ's status shown by the corresponding INTC_IRQPn[PENDING] bit depends on when the IRQ Pending register is read. Before the ISR is entered, the PENDING bit shows that the IRQ is pending. After the ISR is entered, the PENDING bit shows that the IRQ is not pending. 0 IRQ pending for this vector number 1 No IRQ pending for this vector number

12.2.27 Control Register (INTC_CTRL)

Address: E300h base + 1Bh offset = E31Bh

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	INT	IPIC		VAB							INT_DIS	1			0	
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	1	1	1	0	0

INTC_CTRL field descriptions

Field	Description
15 INT	Interrupt This bit reflects the state of the interrupt to the core. 0 No interrupt is being sent to the core. 1 An interrupt is being sent to the core.
14–13 IPIC	Interrupt Priority Level These bits reflect the new interrupt priority level bits being sent to the Core. These bits indicate the priority level needed for a new IRQ to interrupt the current interrupt being sent to the Core. This field is only updated when the core jumps to a new interrupt service routine. 00 Required nested exception priority levels are 0, 1, 2, or 3. 01 Required nested exception priority levels are 1, 2, or 3. 10 Required nested exception priority levels are 2 or3. 11 Required nested exception priority level is 3.
12–6 VAB	Vector number This field shows bits [7:1] of the Vector Address Bus used at the time the last IRQ was taken. In the case of a fast interrupt, it shows the lower address bits of the jump address. This field is only updated when the core jumps to a new interrupt service routine.
5 INT_DIS	Interrupt disable This bit allows the user to disable all interrupts. 0 Normal operation. (default) 1 All interrupts disabled.
4–2 Reserved	This field is reserved. This read-only field is reserved and always has the value 1.
Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

12.3 Functional Description

The Interrupt Controller is a slave on the IPS bus. It contains registers that allow each of the interrupt sources to be set to one of four priority levels (excluding certain interrupts that have fixed priority). All of the interrupt requests of a given level are priority encoded to determine the lowest numerical value of the active interrupt requests for that level. Within a given priority level, number 0 is the highest priority, and number *n*-1 (where *n* is the total number of interrupt sources) is the lowest priority.

12.3.1 Normal Interrupt Handling

After the INTC determines that an interrupt is to be serviced and which interrupt has the highest priority, an interrupt vector address is generated. Normal interrupt handling concatenates the vector base address (VBA) and the vector number to determine the vector address. In this way, an offset into the vector table is generated for each interrupt.

12.3.2 Interrupt Nesting

Interrupt exceptions may be nested to allow the servicing of an IRQ with higher priority than the current exception. The core controls the masking of interrupt priority levels by setting the I0 and I1 bits in its status register (SR).

Table 12-1. Interrupt Mask Bit Settings in Core Status Register

I1 (SR[9])	I0 (SR[8])	Exceptions Permitted	Exceptions Masked
0	0	Priorities 0, 1, 2, 3	None
0	1	Priorities 1, 2, 3	Priority 0
1	0	Priorities 2, 3	Priorities 0, 1
1	1	Priority 3	Priorities 0, 1, 2

The IPIC field of the INTC module's CTRL register reflects the state of the priority level that is presented to the core.

Table 12-2. Interrupt Priority Level Field Settings

IPIC	Current Interrupt Priority Level	Required Nested Exception Priority
00	No interrupt or SWILP	Priorities 0, 1, 2, 3
01	Priority 0	Priorities 1, 2, 3
10	Priority 1	Priorities 2, 3

Table continues on the next page...

Table 12-2. Interrupt Priority Level Field Settings (continued)

IPIC	Current Interrupt Priority Level	Required Nested Exception Priority
11	Priorities 2 or 3	Priority 3

12.3.3 Fast Interrupt Handling

Fast interrupt processing is described in section 9.3.2.2 of the *DSP56800E Core Reference Manual*. The Interrupt Controller recognizes fast interrupts before the core does.

A fast interrupt is defined (to the INTC) by:

1. Setting the priority of the interrupt as level 2, using the appropriate field in the IPR registers.
2. Setting the FIMn register to the appropriate vector number.
3. Setting the FIVALn and FIVAHn registers with the address of the code for the fast interrupt.

When an interrupt occurs, its vector number is compared with the FIM0 and FIM1 register values. If a match occurs, and if the interrupt priority is level 2, the INTC handles the interrupt as a fast interrupt. The INTC takes the vector address from the appropriate FIVALn and FIVAHn registers, instead of generating an address that is an offset from the vector base address (VBA).

The core then fetches the instruction from the indicated vector address. If the instruction is not a JSR, the core starts its fast interrupt handling.

12.4 Interrupts

This module does not produce interrupts.

Chapter 13
DMA Controller

13.1 Introduction

NOTE

For the chip-specific implementation details of this module's instances, see the chip configuration information.

Information found here describes the direct memory access (DMA) controller module. It provides an overview of the module and describes in detail its signals and programming model.

The latter sections of this chapter describe operations, features, and supported data transfer modes in detail.

An example of using several features of the DMA module is described in AN4631: Using the Asynchronous DMA features of the Kinetis L Series.

Note

The designation *n* is used throughout this section to refer to registers or signals associated with one of the four identical DMA channels: DMA0, DMA1, DMA2, or DMA3.

13.1.1 Overview

The DMA controller module enables fast transfers of data, providing an efficient way to move blocks of data with minimal processor interaction. The DMA module, shown in the following figure, has four channels that allow 8-bit, 16-bit, or 32-bit data transfers. Each channel has a dedicated Source Address register (SAR*n*), Destination Address register (DAR*n*), Status register (DSR*n*), Byte Count register (BCR*n*), and Control register (DCR*n*). Collectively, the combined program-visible registers associated with each channel define a transfer control descriptor (TCD). All transfers are dual address, moving

data from a source memory location to a destination memory location with the module operating as a 32-bit bus master connected to the system bus. The programming model is accessed through a 32-bit connection with the slave peripheral bus. DMA data transfers may be explicitly initiated by software or by peripheral hardware requests.

The following figure is a simplified block diagram of the 4-channel DMA controller.

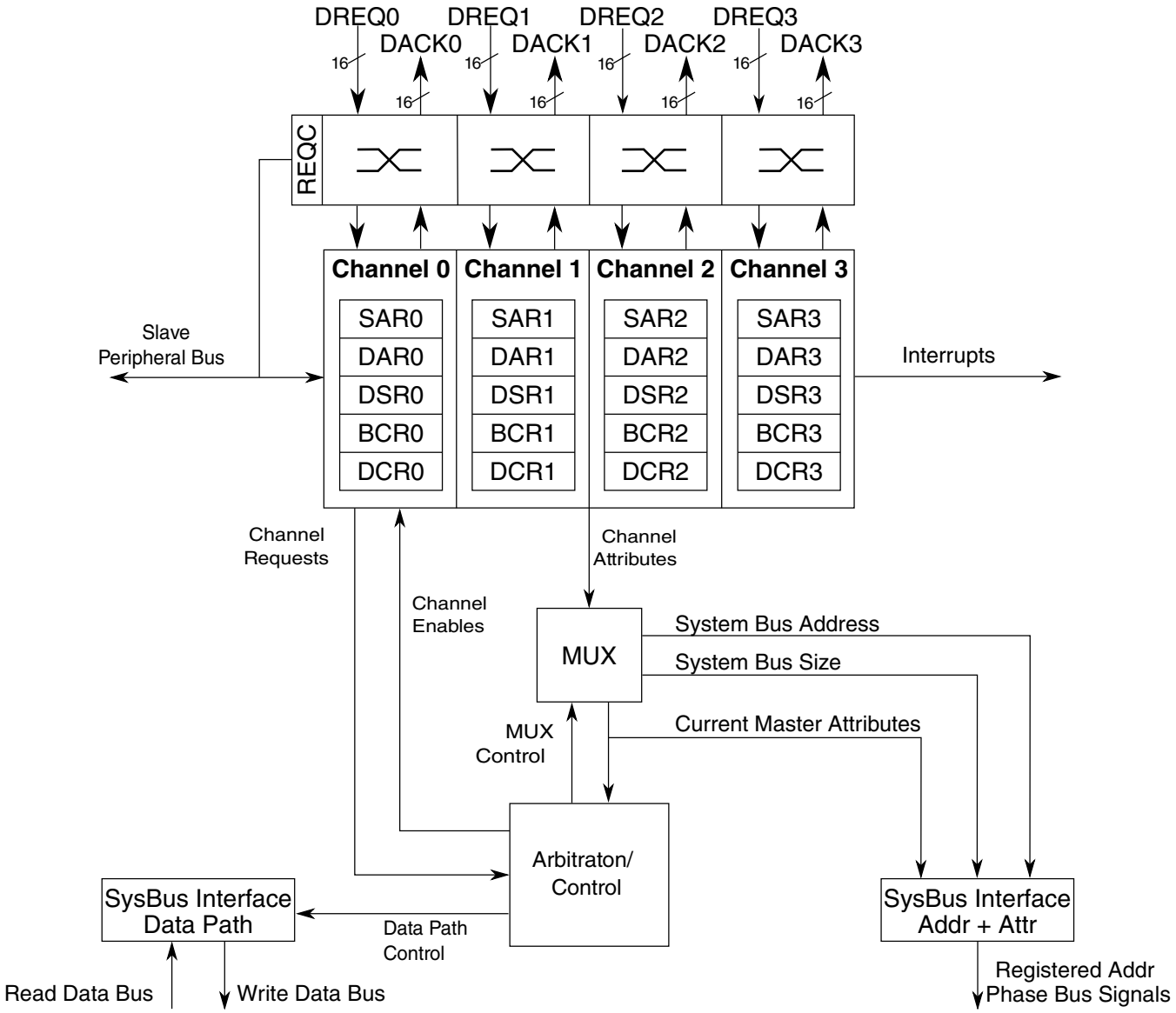


Figure 13-1. 4-Channel DMA Block Diagram

The terms *peripheral request* and *DREQ* refer to a DMA request from one of the on-chip peripherals or package pins. The DMA provides hardware handshake signals: either a DMA acknowledge (DACK) or a done indicator back to the peripheral.For details on the connections associated with DMA request inputs, see the register definition for DMA Request Control (DMAREQC).

13.1.2 Features

The DMA controller module features:

- Four independently programmable DMA controller channels
- Dual-address transfers via 32-bit master connection to the system bus
- Data transfers in 8-, 16-, or 32-bit blocks
- Continuous-mode or cycle-steal transfers from software or peripheral initiation
- One programmable input selected from 16 possible peripheral requests per channel
- Automatic hardware acknowledge/done indicator from each channel
- Independent source and destination address registers
- Optional modulo addressing and automatic updates of source and destination addresses
- Independent transfer sizes for source and destination
- Optional auto-alignment feature for source or destination accesses
- Optional automatic single or double channel linking
- Programming model accessed via 32-bit slave peripheral bus
- Channel arbitration on transfer boundaries using fixed priority scheme

13.2 DMA Transfer Overview

The DMA module can move data within system memory (including memory and peripheral devices) with minimal processor intervention, greatly improving overall system performance.

The DMA module consists of four independent, functionally equivalent channels, so references to DMA in this chapter apply to any of the channels. It is not possible to address all four channels at once.

As soon as a channel has been initialized, it may be started by setting DCR*n*[START] or a properly-selected peripheral DMA request, depending on the status of DCR*n*[ERQ]. Each channel can be programmed to select one peripheral request from a set of 16 possible request inputs.

The DMA controller supports dual-address transfers using its bus master connection to the system bus. The DMA channels support transfers up to 32 data bits in size and have the same memory map addressibility as the processor.

- Dual-address transfers—A dual-address transfer consists of a read followed by a write and is initiated by a request using the DCRn[START] bit or by a peripheral DMA request. The read data is temporarily held in the DMA channel hardware until the write operation. Two types of single transfers occur: a read from a source address followed by a write to a destination address. See the following figure.

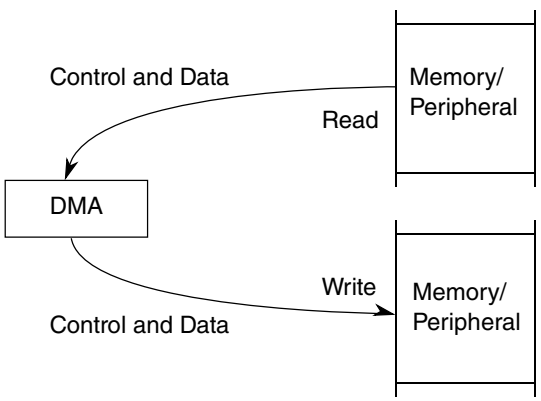


Figure 13-2. Dual-Address Transfer

Any operation involving a DMA channel follows the same three steps:

1. Channel initialization—The transfer control descriptor, contained in the channel registers, is loaded with address pointers, a byte-transfer count, and control information using accesses from the slave peripheral bus.
2. Data transfer—The DMA accepts requests for data transfers. Upon receipt of a request, it provides address and bus control for the transfers via its master connection to the system bus and temporary storage for the read data. The channel performs one or more source read and destination write data transfers.
3. Channel termination—Occurs after the operation is finished successfully or due to an error. The channel indicates the operation status in the channel's DSR, described in the definitions of the DMA Status Registers (DSRn) and Byte Count Registers (BCRn).

13.3 Memory Map/Register Definition

Information about the registers related to the DMA controller module can be found here.

Descriptions of each register and its bit assignments follow. Modifying DMA control registers during a transfer can result in undefined operation. The following table shows the mapping of DMA controller registers. The DMA programming model is accessed via the slave peripheral bus. The concatenation of the source and destination address registers, the status and byte count register, and the control register create a 128-bit transfer control descriptor (TCD) that defines the operation of each DMA channel.

NOTE

On this chip, these registers are addressed in 16-bit words. The base address and offsets are presented in 16-bit words.

DMA memory map

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
C800	DMA Request Control Register (DMA_REQC)	32	R/W	0000_0000h	13.3.1/254
C880	Source Address Register (DMA_SAR0)	32	R/W	0000_0000h	13.3.2/258
C882	Destination Address Register (DMA_DAR0)	32	R/W	0000_0000h	13.3.3/258
C884	DMA Status Register / Byte Count Register (DMA_DSR_BCR0)	32	R/W	0000_0000h	13.3.4/258
C886	DMA Control Register (DMA_DCR0)	32	R/W	0000_0000h	13.3.5/261
C888	Source Address Register (DMA_SAR1)	32	R/W	0000_0000h	13.3.2/258
C88A	Destination Address Register (DMA_DAR1)	32	R/W	0000_0000h	13.3.3/258
C88C	DMA Status Register / Byte Count Register (DMA_DSR_BCR1)	32	R/W	0000_0000h	13.3.4/258
C88E	DMA Control Register (DMA_DCR1)	32	R/W	0000_0000h	13.3.5/261
C890	Source Address Register (DMA_SAR2)	32	R/W	0000_0000h	13.3.2/258
C892	Destination Address Register (DMA_DAR2)	32	R/W	0000_0000h	13.3.3/258
C894	DMA Status Register / Byte Count Register (DMA_DSR_BCR2)	32	R/W	0000_0000h	13.3.4/258
C896	DMA Control Register (DMA_DCR2)	32	R/W	0000_0000h	13.3.5/261
C898	Source Address Register (DMA_SAR3)	32	R/W	0000_0000h	13.3.2/258
C89A	Destination Address Register (DMA_DAR3)	32	R/W	0000_0000h	13.3.3/258
C89C	DMA Status Register / Byte Count Register (DMA_DSR_BCR3)	32	R/W	0000_0000h	13.3.4/258
C89E	DMA Control Register (DMA_DCR3)	32	R/W	0000_0000h	13.3.5/261

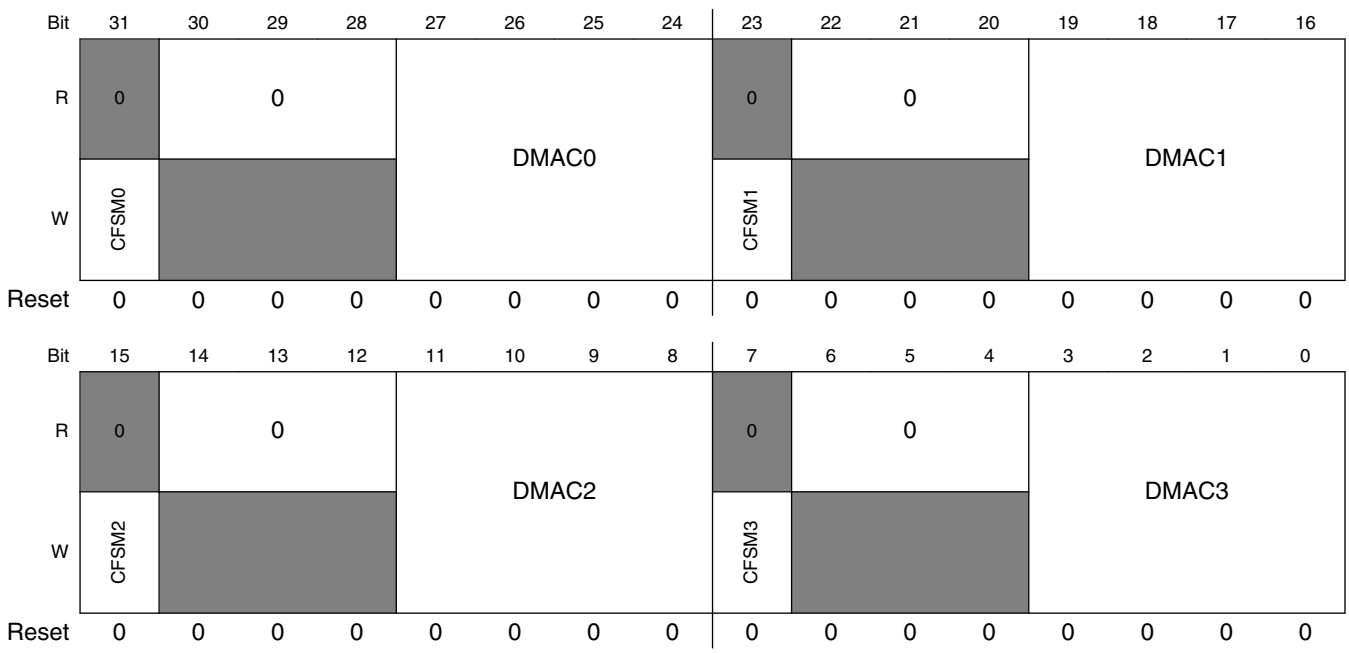
13.3.1 DMA Request Control Register (DMA_REQC)

This register provides a software-controlled connection matrix for DMA requests and acknowledges. Each channel supports 16 possible peripheral requests. The register is programmed to select one peripheral request from the available sources for each channel of the DMA controller. Additionally, the register routes a DMA acknowledge from the channel back to the appropriate peripheral. Writing to this register determines the exact routing of the DMA requests to each of the four channels of the DMA module.

If DCRn[ERQ] is set and the channel is idle, the assertion of the appropriate DREQn signal activates channel n.

The connections of the DMA request sources to the specific channels are device-specific. Refer to the Chip Configuration details for more information.

Address: C800h base + 0h offset = C800h



DMA_REQC field descriptions

Field	Description
31 CFSM0	Clear state machine control 0 This bit clears the state machine for DMA channel 0. When changing the DMAC0 field to select a different requester, set (write 1) to the CFSM0 bit to clear the channel's state machine. Writing 0 to this bit has no effect. The bit always reads as 0. 0 No effect 1 Clear state machine for DMA channel 0

Table continues on the next page...

DMA_REQC field descriptions (continued)

Field	Description
30–28 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
27–24 DMAC0	DMA channel 0 This four-bit field defines the logical connection between the DMA requesters and DMA channel 0. There are sixteen possible requesters per channel and any request from the possible sources can be routed to the DMA channel 0. Effectively, the DMAREQC register provides a software-controlled routing matrix of the DMA request signals to the 4 channels of the DMA module. DMAC0 controls DMA channel 0. The DMA also uses this register to control the broadcasting of acknowledge/done signals back to the selected peripheral to complete the hardware-initiated data transfer. The definition of the 16 possible DMA request sources for each channel is device specific. Refer to the Chip Configuration details for more information. 0000 Select request 0 as the source 0001 Select request 1 as the source 0010 Select request 2 as the source 0011 Select request 3 as the source 0100 Select request 4 as the source 0101 Select request 5 as the source 0110 Select request 6 as the source 0111 Select request 7 as the source 1000 Select request 8 as the source 1001 Select request 9 as the source 1010 Select request 10 as the source 1011 Select request 11 as the source 1100 Select request 12 as the source 1101 Select request 13 as the source 1110 Select request 14 as the source 1111 Select request 15 as the source
23 CFSM1	Clear state machine control 1 This bit clears the state machine for DMA channel 1. When changing the DMAC1 field to select a different requester, set (write 1) to the CFSM1 bit to clear the channel's state machine. Writing 0 to this bit has no effect. The bit always reads as 0. 0 No effect 1 Clear state machine for DMA channel 1
22–20 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
19–16 DMAC1	DMA channel 1 This four-bit field defines the logical connection between the DMA requesters and DMA channel 1. There are sixteen possible requesters per channel and any request from the possible sources can be routed to the DMA channel 1. Effectively, the DMAREQC register provides a software-controlled routing matrix of the DMA request signals to the 4 channels of the DMA module. DMAC1 controls DMA channel 1. The DMA also uses this register to control the broadcasting of acknowledge/done signals back to the selected peripheral to complete the hardware-initiated data transfer. The definition of the 16 possible DMA request sources for each channel is device specific. Refer to the Chip Configuration details for more information.

Table continues on the next page...

DMA_REQC field descriptions (continued)

Field	Description
	0000 Select request 0 as the source 0001 Select request 1 as the source 0010 Select request 2 as the source 0011 Select request 3 as the source 0100 Select request 4 as the source 0101 Select request 5 as the source 0110 Select request 6 as the source 0111 Select request 7 as the source 1000 Select request 8 as the source 1001 Select request 9 as the source 1010 Select request 10 as the source 1011 Select request 11 as the source 1100 Select request 12 as the source 1101 Select request 13 as the source 1110 Select request 14 as the source 1111 Select request 15 as the source
15 CFSM2	Clear state machine control 2 This bit clears the state machine for DMA channel 2. When changing the DMAC2 field to select a different requester, set (write 1) to the CFSM2 bit to clear the channel's state machine. Writing 0 to this bit has no effect. The bit always reads as 0. 0 No effect 1 Clear state machine for DMA channel 2
14–12 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
11–8 DMAC2	DMA channel 2 This four-bit field defines the logical connection between the DMA requesters and DMA channel 2. There are sixteen possible requesters per channel and any request from the possible sources can be routed to the DMA channel 2. Effectively, the DMAREQC register provides a software-controlled routing matrix of the DMA request signals to the 4 channels of the DMA module. DMAC2 controls DMA channel 2. The DMA also uses this register to control the broadcasting of acknowledge/done signals back to the selected peripheral to complete the hardware-initiated data transfer. The definition of the 16 possible DMA request sources for each channel is device specific. Refer to the Chip Configuration details for more information. 0000 Select request 0 as the source 0001 Select request 1 as the source 0010 Select request 2 as the source 0011 Select request 3 as the source 0100 Select request 4 as the source 0101 Select request 5 as the source 0110 Select request 6 as the source 0111 Select request 7 as the source 1000 Select request 8 as the source 1001 Select request 9 as the source 1010 Select request 10 as the source 1011 Select request 11 as the source

Table continues on the next page...

DMA_REQC field descriptions (continued)

Field	Description
	1100 Select request 12 as the source 1101 Select request 13 as the source 1110 Select request 14 as the source 1111 Select request 15 as the source
7 CFSM3	Clear state machine control 3 This bit clears the state machine for DMA channel 3. When changing the DMAC3 field to select a different requester, set (write 1) to the CFSM3 bit to clear the channel's state machine. Writing 0 to this bit has no effect. The bit always reads as 0. 0 No effect 1 Clear state machine for DMA channel 3
6–4 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
DMAC3	DMA channel 3 This four-bit field defines the logical connection between the DMA requesters and DMA channel 3. There are sixteen possible requesters per channel and any request from the possible sources can be routed to the DMA channel 3. Effectively, the DMAREQC register provides a software-controlled routing matrix of the DMA request signals to the 4 channels of the DMA module. DMAC3 controls DMA channel 3. The DMA also uses this register to control the broadcasting of acknowledge/done signals back to the selected peripheral to complete the hardware-initiated data transfer. The definition of the 16 possible DMA request sources for each channel is device specific. Refer to the Chip Configuration details for more information. 0000 Select request 0 as the source 0001 Select request 1 as the source 0010 Select request 2 as the source 0011 Select request 3 as the source 0100 Select request 4 as the source 0101 Select request 5 as the source 0110 Select request 6 as the source 0111 Select request 7 as the source 1000 Select request 8 as the source 1001 Select request 9 as the source 1010 Select request 10 as the source 1011 Select request 11 as the source 1100 Select request 12 as the source 1101 Select request 13 as the source 1110 Select request 14 as the source 1111 Select request 15 as the source

13.3.2 Source Address Register (DMA_SAR*n*)

Address: C800h base + 80h offset + (8d × i), where i=0d to 3d

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	SAR																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

DMA_SAR*n* field descriptions

Field	Description
SAR	SAR Each SAR contains the byte address used by the DMA controller to read data. The SARn is typically aligned on a 0-modulo-ssize boundary—that is, on the natural alignment of the source data. **NOTE:** Most peripheral modules on this chip are addressed in terms on 16-bit words. However, the DMA controller expects a byte address for every module. As a result, in the SARn and DARn registers, the user must always specify a byte address (which is double the value of a 16-bit word address).

13.3.3 Destination Address Register (DMA_DAR*n*)

Address: C800h base + 82h offset + (8d × i), where i=0d to 3d

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	DAR																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

DMA_DAR*n* field descriptions

Field	Description
DAR	DAR Each DAR contains the byte address used by the DMA controller to write data. The DARn is typically aligned on a 0-modulo-dsize boundary—that is, on the natural alignment of the destination data. **NOTE:** Most peripheral modules on this chip are addressed in terms on 16-bit words. However, the DMA controller expects a byte address for every module. As a result, in the SARn and DARn registers, the user must always specify a byte address (which is double the value of a 16-bit word address).

13.3.4 DMA Status Register / Byte Count Register (DMA_DSR_BCR*n*)

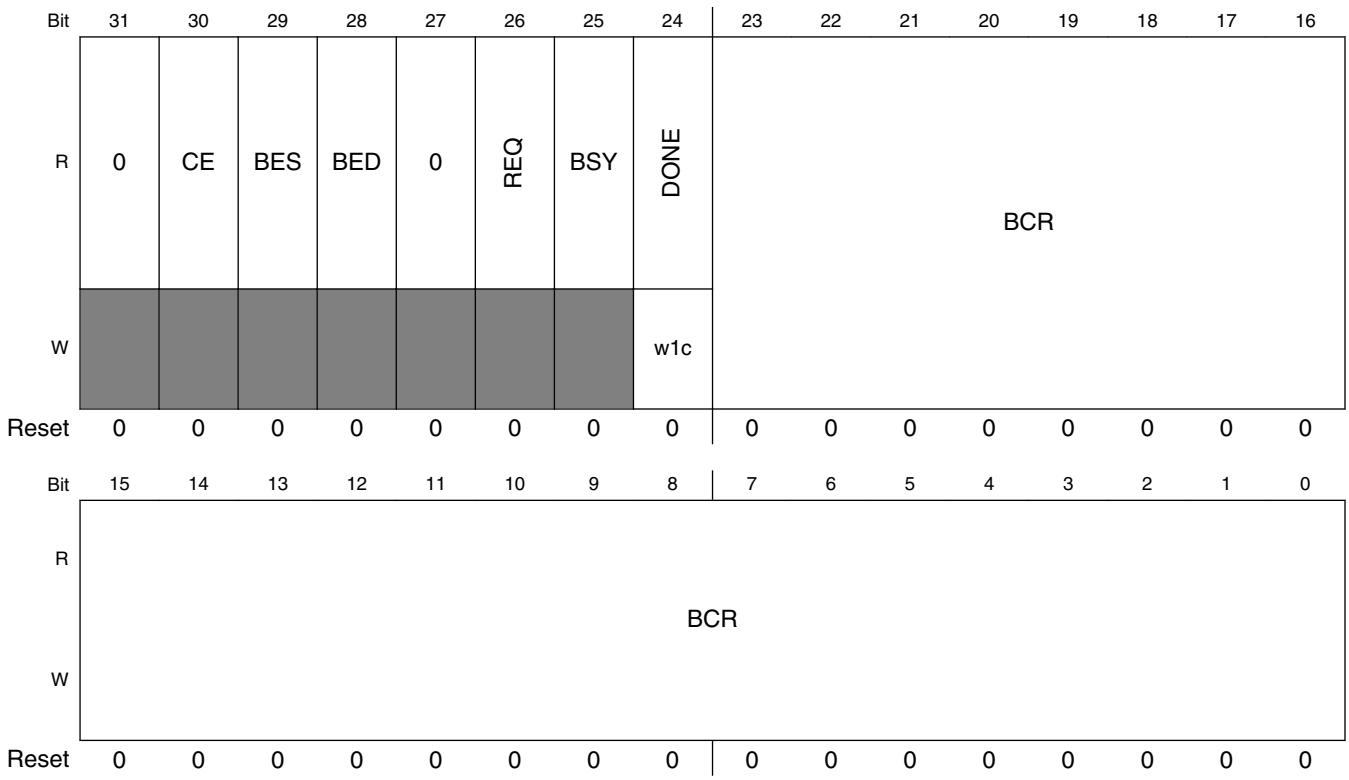
DSR and BCR are two logical registers that occupy one 32-bit address. DSRn occupies bits 31–24, and BCRn occupies bits 23–0. DSRn contains flags indicating the channel status, and BCRn contains the number of bytes yet to be transferred for a given block.

On the successful completion of the write transfer, BCRn decrements by 1, 2, or 4 for 8-bit, 16-bit, or 32-bit accesses, respectively. BCRn is cleared if a 1 is written to DSR[DONE].

In response to an event, the DMA controller writes to the appropriate DSRn bit. Only a write to DSRn[DONE] results in action. DSRn[DONE] is set when the block transfer is complete.

When a transfer sequence is initiated and BCRn[BCR] is not a multiple of 4 or 2 when the DMA is configured for 32-bit or 16-bit transfers, respectively, DSRn[CE] is set and no transfer occurs.

Address: C800h base + 84h offset + (8d × i), where i=0d to 3d



DMA_DSR_BCR*n* field descriptions

Field	Description
31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
30 CE	Configuration Error Any of the following conditions causes a configuration error: • BCR, SAR, or DAR does not match the requested transfer size. • SSIZE or DSIZE is set to an unsupported value. • BCR equals 0 when the DMA receives a start condition. CE is cleared at hardware reset or by writing a 1 to DONE.

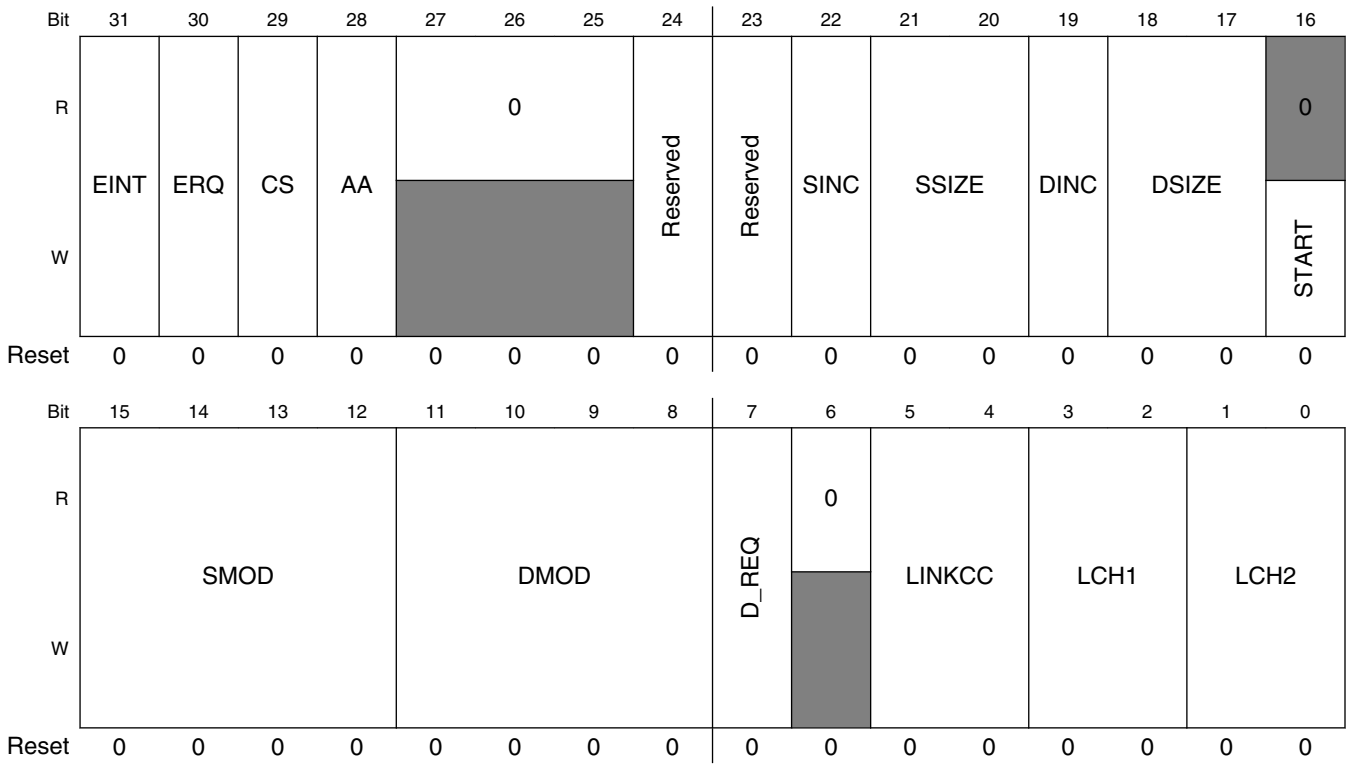
Table continues on the next page...

DMA_DSR_BCR*n* field descriptions (continued)

Field	Description
	0 No configuration error exists. 1 A configuration error has occurred.
29 BES	Bus Error on Source BES is cleared at hardware reset or by writing a 1 to DONE. 0 No bus error occurred. 1 The DMA channel terminated with a bus error during the read portion of a transfer.
28 BED	Bus Error on Destination BED is cleared at hardware reset or by writing a 1 to DONE. 0 No bus error occurred. 1 The DMA channel terminated with a bus error during the write portion of a transfer.
27 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
26 REQ	Request 0 No request is pending or the channel is currently active. Cleared when the channel is selected. 1 The DMA channel has a transfer remaining and the channel is not selected.
25 BSY	Busy 0 DMA channel is inactive. Cleared when the DMA has finished the last transaction. 1 BSY is set the first time the channel is enabled after a transfer is initiated.
24 DONE	Transactions Done Set when all DMA controller transactions complete as determined by transfer count, or based on error conditions. When BCR reaches 0, DONE is set when the final transfer completes successfully. DONE can also be used to abort a transfer by resetting the status bits. When a transfer completes, software must clear DONE before reprogramming the DMA. 0 DMA transfer is not yet complete. Writing a 0 has no effect. 1 DMA transfer completed. Writing a 1 to this bit clears all DMA status bits and should be used in an interrupt service routine to clear the DMA interrupt and error bits.
BCR	BCR This field contains the number of bytes yet to be transferred for a given block.

13.3.5 DMA Control Register (DMA_DCR*n*)

Address: C800h base + 86h offset + (8d × i), where i=0d to 3d



DMA_DCR*n* field descriptions

Field	Description
31 EINT	Enable Interrupt on Completion of Transfer Determines whether an interrupt is generated by completing a transfer or by the occurrence of an error condition. 0 No interrupt is generated. 1 Interrupt signal is enabled.
30 ERQ	Enable Peripheral Request **CAUTION:** Be careful: a collision can occur between START and D_REQ when ERQ is 1. 0 Peripheral request is ignored. 1 Enables peripheral request, defined by the appropriate REQC[DMACn] field, to initiate transfer. A software-initiated request (setting START) is always enabled.
29 CS	Cycle Steal 0 DMA continuously makes read/write transfers until the BCR decrements to 0. 1 Forces a single read/write transfer per request.
28 AA	Auto-align

Table continues on the next page...

DMA_DCR*n* field descriptions (continued)

Field	Description
	AA and SIZE bits determine whether the source or destination is auto-aligned; that is, transfers are optimized based on the address and size. 0 Auto-align disabled 1 If SSIZE indicates a transfer no smaller than DSIZE, source accesses are auto-aligned; otherwise, destination accesses are auto-aligned. Source alignment takes precedence over destination alignment. If auto-alignment is enabled, the appropriate address register increments, regardless of DINC or SINC.
27–25 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
24 Reserved	This field is reserved. **CAUTION:** Must be written as zero; otherwise, undefined behavior results.
23 Reserved	This field is reserved. **CAUTION:** Must be written as zero; otherwise, undefined behavior results.
22 SINC	Source Increment Controls whether the source address increments after each successful transfer. 0 No change to SAR after a successful transfer. 1 The SAR increments by 1, 2, 4 as determined by the transfer size.
21–20 SSIZE	Source Size Determines the data size of the source bus cycle for the DMA controller. 00 32-bit 01 8-bit 10 16-bit 11 Reserved (generates a configuration error (DSRn[CE]) if incorrectly specified at time of channel activation)
19 DINC	Destination Increment Controls whether the destination address increments after each successful transfer. 0 No change to the DAR after a successful transfer. 1 The DAR increments by 1, 2, 4 depending upon the size of the transfer.
18–17 DSIZE	Destination Size Determines the data size of the destination bus cycle for the DMA controller. 00 32-bit 01 8-bit 10 16-bit 11 Reserved (generates a configuration error (DSRn[CE]) if incorrectly specified at time of channel activation)
16 START	Start Transfer 0 DMA inactive 1 The DMA begins the transfer in accordance to the values in the TCDn. START is cleared automatically after one module clock and always reads as logic 0.

Table continues on the next page...

DMA_DCR*n* field descriptions (continued)

Field	Description
15–12 SMOD	Source Address Modulo Defines the size of the source data circular buffer used by the DMA Controller. If enabled (SMOD is non-zero), the buffer base address is located on a boundary of the buffer size. The value of this boundary is based upon the initial source address (SAR). The base address should be aligned to a 0-modulo-(circular buffer size) boundary. Misaligned buffers are not possible. The boundary is forced to the value determined by the upper address bits in the field selection. 0000 Buffer disabled 0001 Circular buffer size is 16 bytes. 0010 Circular buffer size is 32 bytes. 0011 Circular buffer size is 64 bytes. 0100 Circular buffer size is 128 bytes. 0101 Circular buffer size is 256 bytes. 0110 Circular buffer size is 512 bytes. 0111 Circular buffer size is 1 KB. 1000 Circular buffer size is 2 KB. 1001 Circular buffer size is 4 KB. 1010 Circular buffer size is 8 KB. 1011 Circular buffer size is 16 KB. 1100 Circular buffer size is 32 KB. 1101 Circular buffer size is 64 KB. 1110 Circular buffer size is 128 KB. 1111 Circular buffer size is 256 KB.
11–8 DMOD	Destination Address Modulo Defines the size of the destination data circular buffer used by the DMA Controller. If enabled (DMOD value is non-zero), the buffer base address is located on a boundary of the buffer size. The value of this boundary depends on the initial destination address (DAR). The base address should be aligned to a 0-modulo-(circular buffer size) boundary. Misaligned buffers are not possible. The boundary is forced to the value determined by the upper address bits in the field selection. 0000 Buffer disabled 0001 Circular buffer size is 16 bytes 0010 Circular buffer size is 32 bytes 0011 Circular buffer size is 64 bytes 0100 Circular buffer size is 128 bytes 0101 Circular buffer size is 256 bytes 0110 Circular buffer size is 512 bytes 0111 Circular buffer size is 1 KB 1000 Circular buffer size is 2 KB 1001 Circular buffer size is 4 KB 1010 Circular buffer size is 8 KB 1011 Circular buffer size is 16 KB 1100 Circular buffer size is 32 KB 1101 Circular buffer size is 64 KB 1110 Circular buffer size is 128 KB 1111 Circular buffer size is 256 KB
7 D_REQ	Disable Request

Table continues on the next page...

DMA_DCR*n* field descriptions (continued)

Field	Description
	DMA hardware automatically clears the corresponding DCRn[ERQ] bit when the byte count register reaches 0. 0 ERQ bit is not affected. 1 ERQ bit is cleared when the BCR is exhausted.
6 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
5–4 LINKCC	Link Channel Control Allows DMA channels to have their transfers linked. The current DMA channel triggers a DMA request to the linked channels (LCH1 or LCH2) depending on the condition described by the LINKCC bits. If not in cycle steal mode (DCRn[CS]=0) and LINKCC equals 01 or 10, no link to LCH1 occurs. If LINKCC equals 01, a link to LCH1 is created after each cycle-steal transfer performed by the current DMA channel is completed. As the last cycle-steal is performed and the BCR reaches zero, then the link to LCH1 is closed and a link to LCH2 is created. 00 No channel-to-channel linking 01 Perform a link to channel LCH1 after each cycle-steal transfer followed by a link to LCH2 after the BCR decrements to 0. 10 Perform a link to channel LCH1 after each cycle-steal transfer 11 Perform a link to channel LCH1 after the BCR decrements to 0.
3–2 LCH1	Link Channel 1 Indicates the DMA channel assigned as link channel 1. The link channel number cannot be the same as the currently executing channel, and generates a configuration error if this is attempted (DSRn[CE] is set). 00 DMA Channel 0 01 DMA Channel 1 10 DMA Channel 2 11 DMA Channel 3
LCH2	Link Channel 2 Indicates the DMA channel assigned as link channel 2. The link channel number cannot be the same as the currently executing channel, and generates a configuration error if this is attempted (DSRn[CE] is set). 00 DMA Channel 0 01 DMA Channel 1 10 DMA Channel 2 11 DMA Channel 3

13.4 Functional Description

In the following discussion, the term DMA request implies that DCR*n*[START] is set, or DCRn[ERQ] is set and then followed by assertion of the properly selected DMA peripheral request. DCR*n*[START] is cleared when the channel is activated.

Before initiating a dual-address access, the DMA module verifies that DCR*n*[SSIZE] and DCR*n*[DSIZE] are consistent with the source and destination addresses. If they are not consistent, the configuration error bit, DSR*n*[CE], is set. If misalignment is detected, no transfer occurs, DSR*n*[CE] is set, and, depending on the DCR configuration, an interrupt event may be issued. If the auto-align bit, DCR*n*[AA], is set, error checking is performed on the appropriate registers.

A read/write transfer sequence reads data from the source address and writes it to the destination address. The number of bytes transferred is the largest of the sizes specified by DCR*n*[SSIZE] and DCR*n*[DSIZE] in the DMA Control Registers (DCRn).

Source and destination address registers (SAR*n* and DAR*n*) can be programmed in the DCR*n* to increment at the completion of a successful transfer.

13.4.1 Transfer requests (Cycle-Steal and Continuous modes)

The DMA channel supports software-initiated or peripheral-initiated requests. A request is issued by setting DCR*n*[START] or when the selected peripheral request asserts and DCRn[ERQ] is set. Setting DCR*n*[ERQ] enables recognition of the peripheral DMA requests. Selecting between cycle-steal and continuous modes minimizes bus usage for either type of request.

- Cycle-steal mode (DCR*n*[CS] = 1)—Only one complete transfer from source to destination occurs for each request. If DCRn[ERQ] is set, the request is peripheral initiated. A software-initiated request is enabled by setting DCR*n*[START].
- Continuous mode (DCR*n*[CS] = 0)—After a software-initiated or peripheral request, the DMA continuously transfers data until BCRn reaches 0. The DMA performs the specified number of transfers, then retires the channel.

In either mode, the crossbar switch performs independent arbitration on each slave port after each transaction.

13.4.2 Channel initialization and startup

Before a data transfer starts, the channel's transfer control descriptor must be initialized with information describing configuration, request-generation method, and pointers to the data to be moved.

13.4.2.1 Channel prioritization

The four DMA channels are prioritized based on number, with channel 0 having highest priority and channel 3 having the lowest, that is, channel 0 > channel 1 > channel 2 > channel 3.

Simultaneous peripheral requests activate the channels based on this priority order. Once activated, a channel runs to completion as defined by DCR*n*[CS] and BCR*n*.

13.4.2.2 Programming the DMA Controller Module

CAUTION

During a channel's execution, writes to programming model registers can corrupt the data transfer. The DMA module itself does not have a mechanism to prevent writes to registers during a channel's execution.

General guidelines for programming the DMA are:

- The REQC register is configured to select the peripheral DMA requests and assign them to the individual DMA channels.
- TCD*n* is initialized.
  - SAR*n* is loaded with the source (read) address. If the transfer is from a peripheral device to memory or to another peripheral, the source address is the location of the peripheral data register. If the transfer is from memory to a peripheral device or to memory, the source address is the starting address of the data block. This can be any appropriately aligned address.
  - DAR*n* is initialized with the destination (write) address. If the transfer is from a peripheral device to memory, or from memory to memory, DAR*n* is loaded with the starting address of the data block to be written. If the transfer is from memory to a peripheral device, or from a peripheral device to a peripheral device, DAR*n* is loaded with the address of the peripheral data register. This address can be any appropriately aligned address.

- SAR*n* and DAR*n* change after each data transfer depending on DCR*n*[SSIZE, DSIZE, SINC, DINC, SMOD, DMOD] and the starting addresses. Increment values can be 1, 2, or 4 for 8-bit, 16-bit, or 32-bit transfers, respectively. If the address register is programmed to remain unchanged, the register is not incremented after the data transfer.
- BCR*n*[BCR] must be loaded with the total number of bytes to be transferred. It is decremented by 1, 2, or 4 at the end of each transfer, depending on the transfer size. DSR*n*[DONE] must be cleared for channel startup.

- After the channel has been initialized, it may be started by setting DCR*n*[START] or a properly selected peripheral DMA request, depending on the status of DCR*n*[ERQ]. For a software-initiated transfer, the channel can be started by setting DCR*n*[START] as part of a single 32-bit write to the last 32 bits of the TCD*n*; that is, it is not required to write the DCR*n* with START cleared and then perform a second write to explicitly set START.
- Programming the channel for a software-initiated request causes the channel to request the system bus and start transferring data immediately. If the channel is programmed for peripheral-initiated request, a properly selected peripheral DMA request must be asserted before the channel begins the system bus transfers.
- The hardware can automatically clear DCR*n*[ERQ], disabling the peripheral request, when BCR*n* reaches zero by setting DCR*n*[D_REQ].
- Changes to DCR*n* are effective immediately while the channel is active. To avoid problems with changing a DMA channel setup, write a one to DSR*n*[DONE] to stop the DMA channel.

13.4.3 Dual-Address Data Transfer Mode

Each channel supports dual-address transfers. Dual-address transfers consist of a source data read and a destination data write. The DMA controller module begins a dual-address transfer sequence after a DMA request. If no error condition exists, DSR*n*[REQ] is set.

- Dual-address read—The DMA controller drives the SAR*n* value onto the system address bus. If DCR*n*[SINC] is set, the SAR*n* increments by the appropriate number of bytes upon a successful read cycle. When the appropriate number of read cycles complete (multiple reads if the destination size is larger than the source), the DMA initiates the write portion of the transfer.

If a termination error occurs, DSR*n*[BES, DONE] are set and DMA transactions stop.

- Dual-address write—The DMA controller drives the DAR*n* value onto the system address bus. When the appropriate number of write cycles complete (multiple writes if the source size is larger than the destination), DAR*n* increments by the appropriate number of bytes if DCRn[DINC] is set. BCRn decrements by the appropriate number of bytes. DSR*n*[DONE] is set when BCR*n* reaches zero. If the BCR*n* is greater than zero, another read/write transfer is initiated if continuous mode is enabled (DCRn[CS] = 0).

  If a termination error occurs, DSR*n*[BED, DONE] are set and DMA transactions stop.

13.4.4 Advanced Data Transfer Controls: Auto-Alignment

Typically, auto-alignment for DMA transfers applies for transfers of large blocks of data. As a result, it does not apply for peripheral-initiated cycle-steal transfers.

Auto-alignment allows block transfers to occur at the optimal size based on the address, byte count, and programmed size. To use this feature, DCR*n*[AA] must be set. The source is auto-aligned if DCR*n*[SSIZE] indicates a transfer size larger than DCR*n*[DSIZE]. Source alignment takes precedence over the destination when the source and destination sizes are equal. Otherwise, the destination is auto-aligned. The address register chosen for alignment increments regardless of the increment value. Configuration error checking is performed on registers not chosen for alignment.

If BCR*n* is greater than 16, the address determines transfer size. Transfers of 8 bits, 16 bits, or 32 bits are transferred until the address is aligned to the programmed size boundary, at which time accesses begin using the programmed size. If BCR*n* is less than 16 at the start of a transfer, the number of bytes remaining dictates transfer size.

Consider this example, which shows byte addresses inside the DMA registers:

- AA equals 1.
- SAR*n* equals 0x0101 (core data address in RAM).
- BCR*n* equals 0x00_00F0.
- SSIZE equals 00 (32 bits).
- DSIZE equals 01 (8 bits).

Because SSIZE > DSIZE, the source is auto-aligned. Error checking is performed on destination registers. The access sequence is as follows:

1. Read 1 byte from 0x0101, increment SAR*n*, write 1 byte (using DAR*n*).
2. Read 2 bytes from 0x0102, increment SAR*n*, write 2 bytes.

3. Read 4 bytes from 0x0104, increment SAR*n*, write 4 bytes.
4. Repeat 4-byte operations until SAR*n* equals 0x01F0.
5. Read byte from 0x01F0, increment SAR*n*, write byte.

If DSIZE is another size, data writes are optimized to write the largest size allowed based on the address, but not exceeding the configured size.

13.4.5 Termination

An unsuccessful transfer can terminate for one of the following reasons:

- Error conditions—When the DMA encounters a read or write cycle that terminates with an error condition, DSR*n*[BES] is set for a read and DSR*n*[BED] is set for a write before the transfer is halted. If the error occurred in a write cycle, data in the internal holding registers is lost.
- Interrupts—If DCR*n*[EINT] is set, the DMA drives the appropriate interrupt request signal. The processor can read DSR*n* to determine whether the transfer terminated successfully or with an error. DSR*n*[DONE] is then written with a 1 to clear the interrupt, DSR*n*[DONE], and error status bits.

Chapter 14
Power Management Controller (PMC)

14.1 Introduction

14.1.1 Overview

This specification details the on-chip power management controller module. This module contains the core voltage regulators and power monitoring circuitry. Its function is to ensure that the chip is operated only within legal voltage ranges and to assist in the orderly shutdown of the chip in the event that the power supply is interrupted. It also regulates the internal voltage rails for the core digital and analog logic.

14.1.2 Features

- Small voltage regulator generates 2.7 V and 1.2 V regulated supply for analog components.
- Large voltage regulator generates 1.2 V regulated supply for core digital logic.
- LVI27 Low voltage interrupt generated when VDDIO drops consistently below 2.7 V.
- LVI22 Low voltage interrupt generated when VDDIO drops consistently below 2.2 V.
- Power On Reset asserts when VDDIO drops below 2.0 V.
- Power on Reset deasserts only when VDDIO is consistently above 2.7 V.
- POR, LVI27, and LVI22 levels have 50-100 mV of internal hysteresis.
- Large regulator has standby mode which reduces its power consumption but limits the current it can supply to the part.
- Flag to indicate when the small regulator's 2.7 V supply is ready to be used.

The assumption is made that power supply voltages move relatively slowly with respect to the system clocks, allowing the chip some control over its future operation.

14.1.3 Modes of Operation

By definition, this module has the most impact on the chip when the power supply voltages are moving. The PMC implements power supply regulation, power on reset, and low voltage detection functions. The internal integration of these functions as well as the integration of power supplies with other system functions is illustrated in the following diagram.

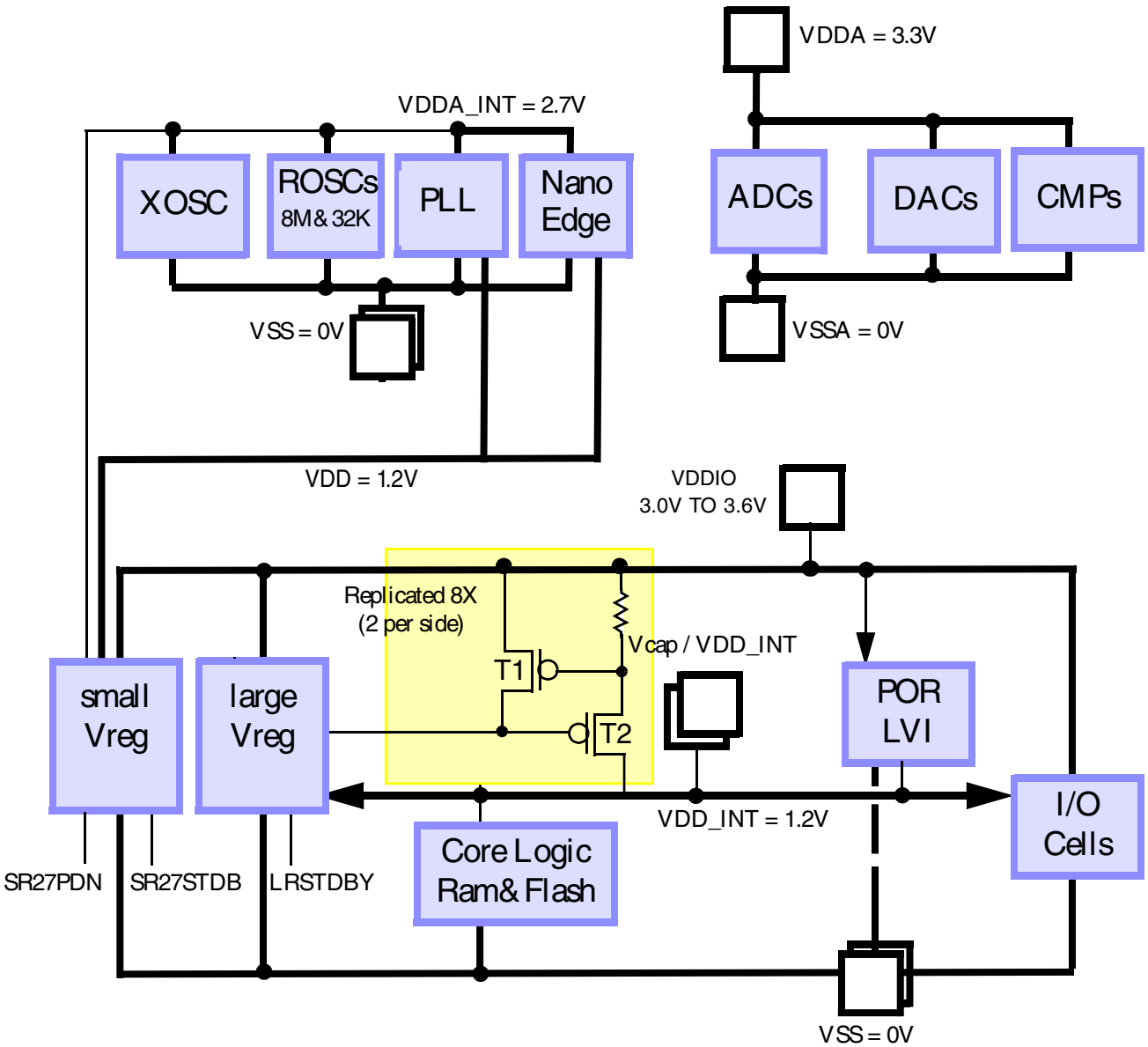


Figure 14-1. Illustrative integration of the Power Management Controller

Figure 14-1 illustrates the integration of the power supervisor. The large regulator generates the 1.2V VDD_INT digital supply which is used by IO cells, core logic, Flash and RAM. The small regulator generates the 2.7V VDDA_INT analog supply which is

used by various analog modules. The POR_LVI module uses the unregulated supply VDDIO to generate a raw POR reset which releases at VDDIO=2.0V, an LV22 low voltage detect which releases at VDDIO=2.2V and an LV27 low voltage detect which releases at VDDIO=2.7V. The PMC uses the POR and low voltage detect signals to provide a flexible infrastructure for detecting VDDIO changes relative to the two LVI detect levels and invoking the low voltage interrupt.

The SR27PDN, SR27STDBY, and LRSTDBY inputs are controlled from memory mapped register fields in the SIM. Standby mode reduces a regulator's drive capacity but also reduces its power consumption. LRSTDBY will control large regulator standby mode. SR27STDBY will control standby mode for the 2.7V supply from the small regulator. SR27PDN will control the powerdown of the 2.7V supply from the small regulator and takes precedence over SR27STDBY.

The standby and powerdown controls for the 1.2 V supply from the small regulator are tied in the powered down state because this supply is unused.

14.1.4 Block Diagram

The internal organization of the PMC is illustrated in Figure 14-2.

NOTE

The deglitch flops must be clocked by a continuously running clock so these interrupts can wake the part up if it is in stop mode.

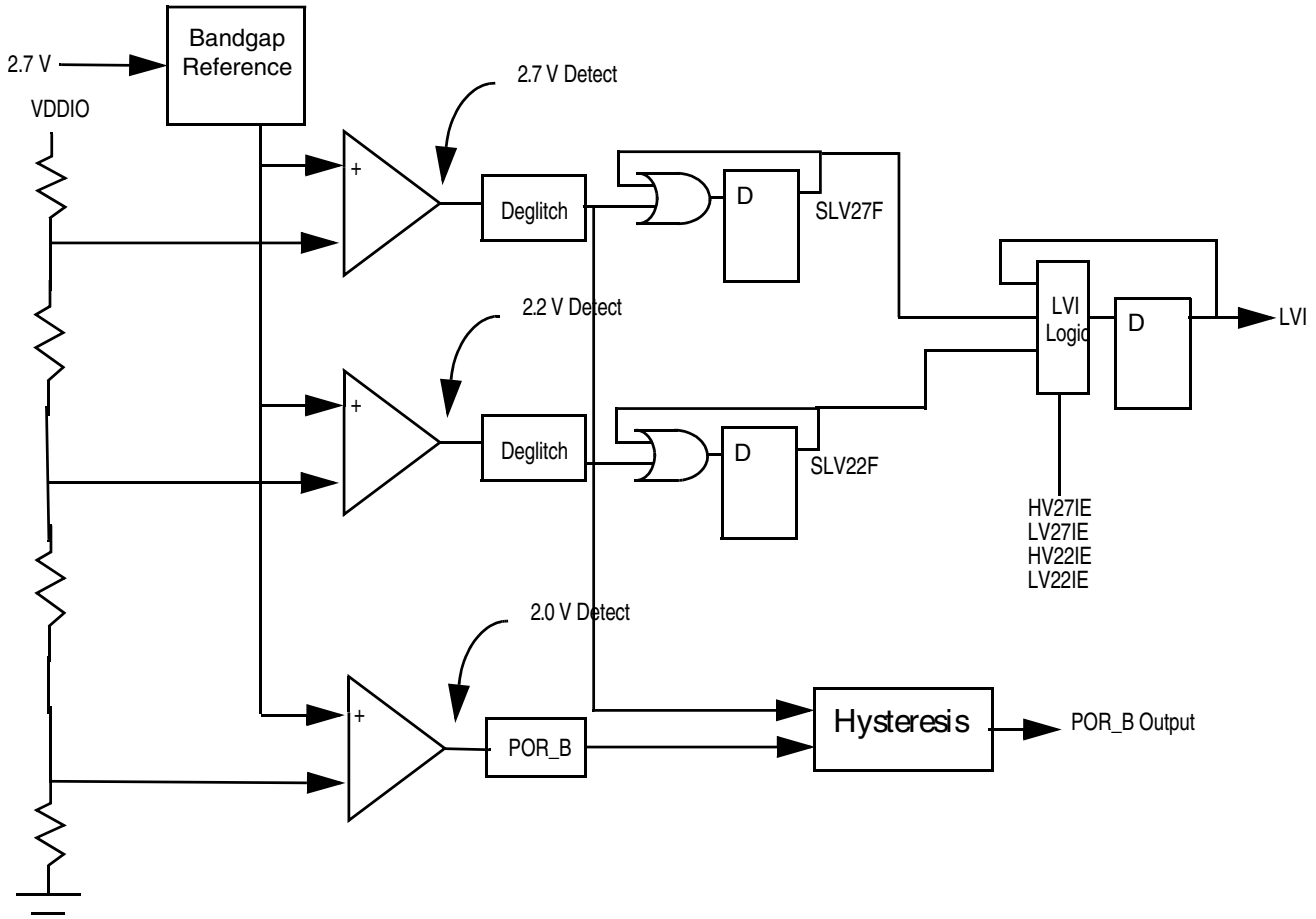


Figure 14-2. PMC Block Diagram

14.2 Memory Map and Register Descriptions

PMC memory map

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
E2A0	Control Register (PMC_CTRL)	16	R/W	7000h	14.2.1/275
E2A1	Status Register (PMC_STS)	16	w1c	0020h	14.2.2/276

14.2.1 Control Register (PMC_CTRL)

Address: E2A0h base + 0h offset = E2A0h

Bit	15	14	13	12	11	10	9	8
Read	TRIM				0			
Write								
Reset	0	1	1	1	0	0	0	0

Bit	7	6	5	4	3	2	1	0
Read	VRBEN	0			HV27IE	HV22IE	LV27IE	LV22IE
Write								
Reset	0	0	0	0	0	0	0	0

PMC_CTRL field descriptions

Field	Description
15–12 TRIM	Bandgap Trim This field is used to trim the bandgap reference in the regulator. Its reset state is mid-range.
11–8 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
7 VRBEN	Voltage Reference Buffer Enable This bit enables a buffer that drives the 1.2 V bandgap reference to the ADC. This bit should be enabled if the user wants to calibrate the ADC using the 1.2 V reference. The bit may be disabled to save power when ADC calibration is not being performed. 0 Disable voltage reference buffering. 1 Enable voltage reference buffering.
6–4 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
3 HV27IE	2.7 V High Voltage Interrupt Enable This bit allows the STS[LVI] bit to be set when the STS[LV27F] bit is clear. While LV27F is high (VDDIO is below 2.7 V), set this bit and an LVI interrupt is generated when LV27F becomes low (VDDIO becomes greater than 2.7 V). 0 Disable setting the high voltage interrupt. 1 Enable setting the high voltage interrupt.
2 HV22IE	2.2 V High Voltage Interrupt Enable This bit allows the STS[LVI] bit to be set when the STS[LV22F] bit is clear. While LV22F is high (VDDIO is below 2.2 V), set this bit and an LVI interrupt is generated when LV22F becomes low (VDDIO becomes greater than 2.2 V). 0 Disable setting the high voltage interrupt. 1 Enable setting the high voltage interrupt.
1 LV27IE	2.7 V Low Voltage Interrupt Enable

Table continues on the next page...

PMC_CTRL field descriptions (continued)

Field	Description
	This bit allows the STS[LVI] bit to be set when the STS[LV27F] bit is set. While LV27F is low (VDDIO is above 2.7 V), set this bit and an LVI interrupt is generated when LV27F becomes high (VDDIO becomes lower than 2.7 V). 0 Disable setting the low voltage interrupt. 1 Enable setting the low voltage interrupt.
0 LV22IE	2.2 V Low Voltage Interrupt Enable This bit allows the STS[LVI] bit to be set when the STS[LV22F] bit is set. While LV22F is low (VDDIO is above 2.2 V), set this bit and an LVI interrupt is generated when LV22F becomes high (VDDIO becomes lower than 2.2 V). 0 Disable setting the low voltage interrupt. 1 Enable setting the low voltage interrupt.

14.2.2 Status Register (PMC_STS)

Address: E2A0h base + 1h offset = E2A1h

Bit	15	14	13	12	11	10	9	8
Read	0							
Write								
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
Read	0		SR27	LVI	SLV27F	SLV22F	LV27F	LV22F
Write				w1c	w1c	w1c		
Reset	0	0	1	0	0	0	0	0

PMC_STS field descriptions

Field	Description
15–6 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
5 SR27	Small Regulator 2.7 V Active Flag This read-only bit indicates that the small regulator 2.7 V supply, which supplies power to the crystal oscillator, relaxation oscillator, PLL, and duty cycle corrector, is powered up and ready to use. The small regulator 2.7 V supply is powered down using the SIM's PWR[SR27PDN] bits. The small regulator requires 100 µs to wake up from power down. Since the maximum clock frequency is 200 kHz while in VLPRUN mode (the only mode where the small regulator is powered down), this flag is set about 20 clock cycles after the SIM's PWR[SR27PDN] bits turn on the small regulator 2.7 V supply. Verify this flag is set before turning on any of the devices that are powered by the small regulator's 2.7 V supply. 0 The small regulator 2.7 V supply is not ready to be used. 1 The small regulator 2.7 V supply is ready to be used.

Table continues on the next page...

PMC_STS field descriptions (continued)

Field	Description
4 LVI	Low Voltage Interrupt This bit is the low voltage interrupt. This bit is set by any of several conditions: • STS[LV22F] and CTRL[LV22IE], • STS[LV27F] and CTRL[LV27IE], • $\overline{\text{STS[LV22F]}}$ and CTRL[HV22IE], or • $\overline{\text{STS[LV27F]}}$ and CTRL[HV27IE] Once set, this bit remains set until a 1 is written to this bit position or a reset occurs. Writing a 0 has no effect. Following is an explanation of how to configure these controls to detect rising and/or falling transitions of VDDIO relative to 2.7 V or 2.2 V. When STS[LV27F] is low (VDDIO is above 2.7 V), both HV22IE and HV27IE should be cleared. Setting LV27IE and clearing LV22IE asserts LVI when VDDIO falls below 2.7 V. Setting LV22IE and clearing LV27F asserts LVI if VDDIO falls below 2.2 V. Setting both LV27F and LV22F has the same effect as setting only LV27F, which asserts LVI when VDDIO falls below 2.7 V. When STS[LV22F] and STS[LV27F] are both high (VDDIO is below 2.2 V), both LV22IE and LV27IE should be cleared. Setting HV22IE and clearing HV27IE asserts LVI when VDDIO rises above 2.2 V. Setting HV27IE and clearing HV22IE asserts LVI if VDDIO rises above 2.7 V. Setting both HV22IE and HV27IE has the same effect as setting only HV22IE, which asserts LVI when VDDIO rises above 2.2 V. When STS[LV27F] is high (VDDIO is below 2.7 V) and STS[LV22F] is low (VDDIO is above 2.2 V), both LV27IE and HV22IE should be cleared. Setting HV27IE and clearing LV22IE asserts LVI only if VDDIO rises above 2.7 V. Setting LV22IE and clearing HV22IE asserts LVI only if VDDIO falls below 2.2 V. Setting both HV27IE and LV22IE asserts LVI if VDDIO either falls below 2.2 V or rises above 2.7 V. 0 Low voltage interrupt cleared. 1 Low voltage interrupt asserted.
3 SLV27F	Sticky 2.7 V Low Voltage Flag This sticky bit indicates that the 3.3 V supply dropped below the 2.7 V level at some point. Once set, this bit remains set until a 1 is written to this bit position or a reset occurs. Writing a 0 has no effect. 0 3.3 V supply has not dropped below the 2.7 V threshold. 1 3.3 V supply has dropped below the 2.7 V threshold.
2 SLV22F	Sticky 2.2 V Low Voltage Flag This sticky bit indicates that the 3.3 V supply dropped below the 2.2 V level at some point. Once set, this bit remains set until a 1 is written to this bit position or a reset occurs. Writing a 0 has no effect. 0 3.3 V supply has not dropped below the 2.2 V threshold. 1 3.3 V supply has dropped below the 2.2 V threshold.
1 LV27F	2.7 V Low Voltage Flag This read-only bit indicates that the 3.3 V supply is currently below the 2.7 V level. This bit may reset itself if the supply voltage rises above the threshold. 0 3.3 V supply is not below the 2.7 V threshold. 1 3.3 V supply is below the 2.7 V threshold.
0 LV22F	2.2 V Low Voltage Flag This read-only bit indicates that the 3.3 V supply is currently below the 2.2 V level. This bit may reset itself if the supply voltage rises above the threshold.

Table continues on the next page...

PMC_STS field descriptions (continued)

Field	Description
	0 3.3 V supply is not below the 2.2 V threshold. 1 3.3 V supply is below the 2.2 V threshold.

14.3 Functional Description

As shown in the block diagram, V_{DDIO} is processed by the POR_LVI analog module to generate an internal power-on reset and LV22 and LV27 low voltage detection signals. The PMC performs deglitch functions on these signals and uses them to generate noise-free versions of the raw POR and low voltage detects. These are available in both raw and sticky (registered) forms.

The LVI logic in the PMC uses four interrupt enables and the two low voltage detects for V_{DDIO} = 2.7 V and V_{DDIO} = 2.2 V to generate a single low voltage interrupt. By properly configuring these four interrupt enables based upon the current V_{DDIO} voltage range (as indicated by two low voltage detect signals), the LVI can be configured to assert on any possible falling or rising transition of VDDIO through either of the two fixed LVI levels.

The POR circuit is designed to assert the internal POR reset until V_{DDIO} is above 2.0 V. The hysteresis function of the PMC, however, keeps the POR_B output asserted until LV27 detection indicates that V_{DDIO} is above 2.7 V. The POR_B output will not reassert until internal V_{DDIO} falls below 2.0 V.

The deglitch blocks are essentially strings of 4 flops in series. The outputs of all 4 must agree before the STS[LV27F or LV22F] bits change state. This is intended to prevent the LVI circuitry from responding to momentary glitches brought about as a result of normal operation.

Figure 14-3 illustrates operation of the POR_B versus low voltage detect circuits. Low voltage detection circuits indicate the voltage on V_{DDIO}, the external 3.3 V supply, relative to 2.7 V and 2.2 V.

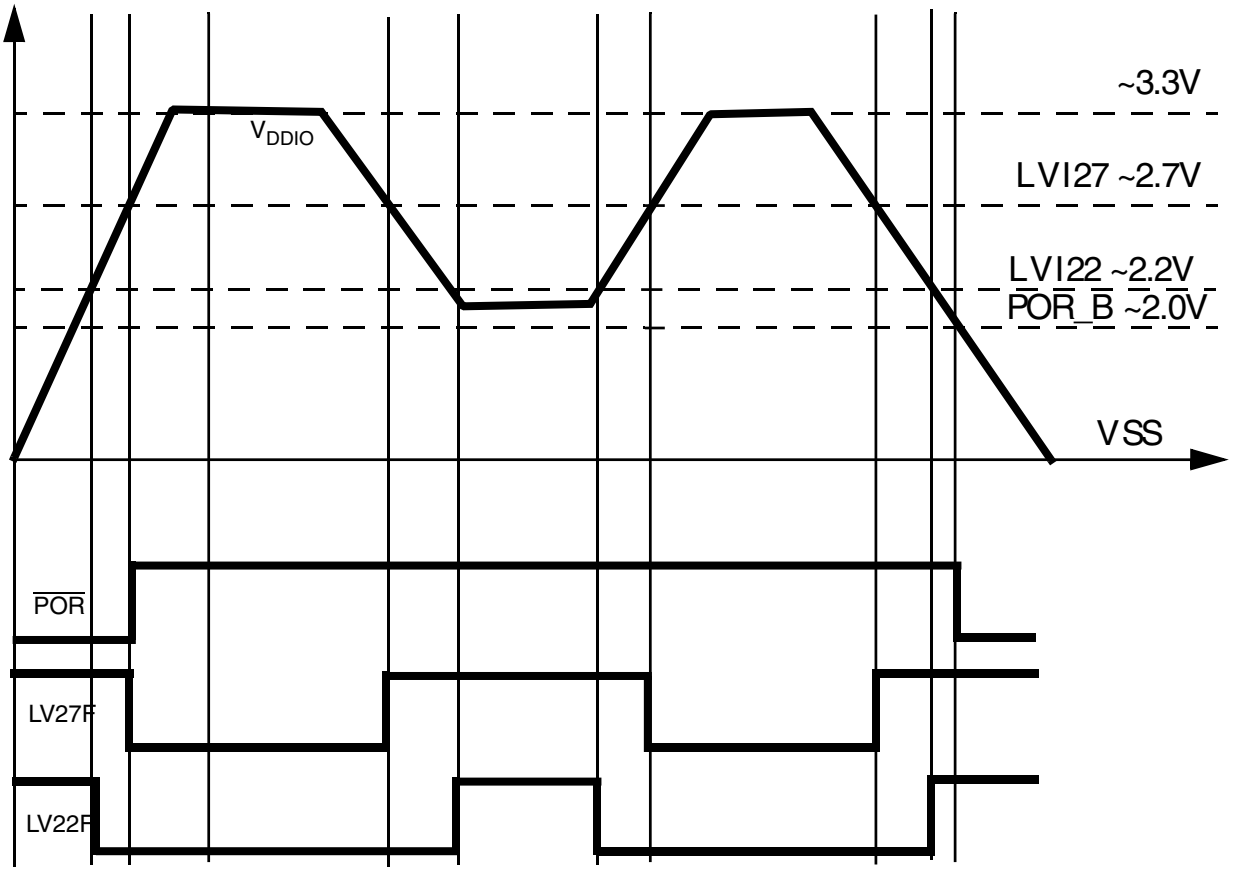


Figure 14-3. POR_B Versus Low-Voltage Detects

14.4 Resets

Table 14-1. Reset Summary

Reset	Source	Characteristics
POR_B	This module	When asserted, indicates that the supply voltage is too low for reliable operation.
RESET_B	SIM	Used to reset the power management controller registers. This signal is usually derived from POR_B and other chip reset sources.

14.5 Clocks

Clock	Source	Used by
IPBus Clock	SIM	Used by glitch filter during normal operation and by control registers at all times.
Oscillator Clock	Oscillator	Used by glitch filter during stop mode and during power on resets.

14.6 Interrupts

Table 14-2. Interrupt Summary

Core Interrupt	Interrupt Flag	Interrupt Enable	Name	Description
ipi_int_lvi (STS[LVI])	STS[SLV27F], STS[SLV22F]	CTRL[LV27IE], CTRL[LV22IE], CTRL[HV27IE], CTRL[HV22IE]	Low Voltage Interrupt	See Memory Map and Register Descriptions for details

Chapter 15
Crossbar AND/OR/INVERT (AOI) Module

15.1 Introduction

The AND/OR/INVERT module (known simply as the AOI module) supports the generation of a configurable number of EVENT signals. Each output EVENTn is a configurable and/or/invert function of four associated AOI inputs: An, Bn, Cn, and Dn.

This module is designed to be integrated in conjuction with one or more inter-peripheral crossbar switch (XBAR) modules. A crossbar switch is typically used to select the 4*n AOI inputs from among available peripheral outputs and GPIO signals. The n EVENTn outputs from the AOI module are typically used as additional inputs to a second crossbar switch, adding to it the ability to connect to its outputs an arbitrary 4-input boolean function of its other inputs.

The AOI controller is a slave peripheral module connecting event input indicators from a variety of device modules and generating event output signals that can be routed to an inter-peripheral crossbar switch or other peripherals. Its programming model is accessed through the standard IPS (Sky Blue) slave interface. The module is designed to be very configurable in terms of the functionality of its integrated AOI functions.

15.1.1 Overview

The AOI module supports a configurable number of event outputs, where each event output represents a user-programmed combinational boolean function based on four event inputs. The key features of this module include:

- Four dedicated inputs for each event output
- User-programmable combinational boolean function evaluation for each event output
- Memory-mapped device connected to a slave peripheral (IPS) bus
- Configurable number of event outputs

NOTE

The connections from the AOI module outputs to other functions is SoC-specific.

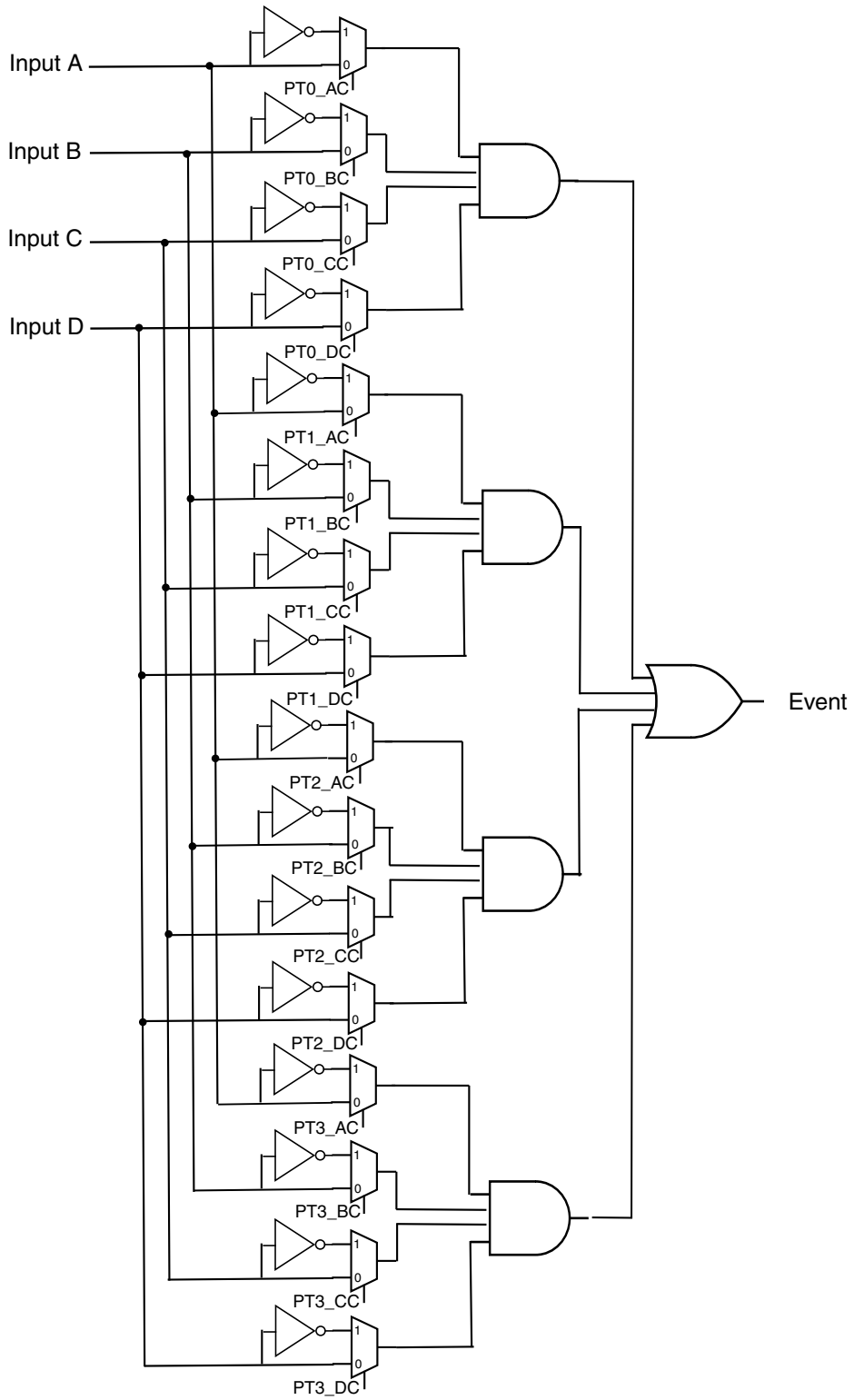


Figure 15-1. Simplified AOI Block Diagram

15.1.2 Features

The major features of the AOI module are summarized below:

- Highly programmable module for creating combinational boolean events for use as hardware triggers
  - Each channel has four event inputs and one output
  - Evaluates a combinational boolean expression as the sum of four products where each product term includes all four selected input sources available as true or complement values
  - Event output is formed as purely combinational logic and operates as a hardware trigger
- Memory-mapped device connected to the slave peripheral (IPS) bus
  - Programming model organized per channel for simplified software

15.1.3 Modes of Operation

The AOI module does not support any special modes of operation. As shown in Figure 15-1, its operation is primarily controlled by the selected event inputs and outputs. Additionally, as a memory-mapped device located on the slave peripheral bus, it responds based strictly on memory address for accesses to its programming model.

The AOI module resides in the slave peripheral *bus clock domain*.

15.2 External Signal Description

The AOI module does not directly support any external interfaces. There may be package input signals (indirectly) connected to the module as event inputs, but since the *AOI does not include any input synchronization hardware*, this function must be handled before the event input signals are routed into the module.

15.3 Memory Map and Register Descriptions

The AOI module supports access to its programming model via a 16-bit peripheral bus connection. The module is designed to support 16-bit accesses only. Functionality for accesses of other widths is undefined.

The AOI module supports a specific number of event outputs. Each output EVENTn outputs a four-term AOI function of four binary inputs: An, Bn, Cn, and Dn. A pair of 16-bit registers configures this four-term AOI function: The two registers BFCRT01n and BFCRT23n define the configuration for the evaluation of the Boolean function defining EVENTn, where *n* is the event output channel number. The BFCRT01n register defines the configuration of product terms 0 and 1, and the BFCRT23n register defines the configuration of product terms 2 and 3.

The AOI module provides a universal Boolean function generator using a four-term sum of products expression with each product term containing true or complement values of the four selected event inputs (An, Bn, Cn, Dn). Specifically, the EVENTn output is defined by the following "4 x 4" Boolean expression:

```
EVENTn
    = (0,An,~An,1) & (0,Bn,~Bn,1) & (0,Cn,~Cn,1) & (0,Dn,~Dn,1)// product term 0
    | (0,An,~An,1) & (0,Bn,~Bn,1) & (0,Cn,~Cn,1) & (0,Dn,~Dn,1)// product term 1
    | (0,An,~An,1) & (0,Bn,~Bn,1) & (0,Cn,~Cn,1) & (0,Dn,~Dn,1)// product term 2
    | (0,An,~An,1) & (0,Bn,~Bn,1) & (0,Cn,~Cn,1) & (0,Dn,~Dn,1)// product term 3
```

where each selected input of each product term can be configured to produce a logical 0 or 1 or pass the true or complement of the selected event input. Each product termuses 8 bits of configuration information, 2 bits for each of the four selected event inputs. The resulting logic provides a simple yet powerful Boolean function evaluation for defining an event output.

These AOI functions are combinational in nature and are intended to be sampled and used synchronously.

AOI memory map

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
E380	Boolean Function Term 0 and 1 Configuration Register for EVENTn (AOI_BFCRT010)	16	R/W	0000h	15.3.1/285
E381	Boolean Function Term 2 and 3 Configuration Register for EVENTn (AOI_BFCRT230)	16	R/W	0000h	15.3.2/286
E382	Boolean Function Term 0 and 1 Configuration Register for EVENTn (AOI_BFCRT011)	16	R/W	0000h	15.3.1/285
E383	Boolean Function Term 2 and 3 Configuration Register for EVENTn (AOI_BFCRT231)	16	R/W	0000h	15.3.2/286
E384	Boolean Function Term 0 and 1 Configuration Register for EVENTn (AOI_BFCRT012)	16	R/W	0000h	15.3.1/285
E385	Boolean Function Term 2 and 3 Configuration Register for EVENTn (AOI_BFCRT232)	16	R/W	0000h	15.3.2/286
E386	Boolean Function Term 0 and 1 Configuration Register for EVENTn (AOI_BFCRT013)	16	R/W	0000h	15.3.1/285
E387	Boolean Function Term 2 and 3 Configuration Register for EVENTn (AOI_BFCRT233)	16	R/W	0000h	15.3.2/286

15.3.1 Boolean Function Term 0 and 1 Configuration Register for EVENTn (AOI_BFCRT01*n*)

Address: E380h base + 0h offset + (2d × i), where i=0d to 3d

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read Write	PT0_AC		PT0_BC		PT0_CC		PT0_DC		PT1_AC		PT1_BC		PT1_CC		PT1_DC	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

AOI_BFCRT01*n* field descriptions

Field	Description
15–14 PT0_AC	Product term 0, A input configuration This 2-bit field defines the Boolean evaluation associated with the selected input A in product term 0. 00 Force the A input in this product term to a logical zero 01 Pass the A input in this product term 10 Complement the A input in this product term 11 Force the A input in this product term to a logical one
13–12 PT0_BC	Product term 0, B input configuration This 2-bit field defines the Boolean evaluation associated with the selected input B in product term 0. 00 Force the B input in this product term to a logical zero 01 Pass the B input in this product term 10 Complement the B input in this product term 11 Force the B input in this product term to a logical one
11–10 PT0_CC	Product term 0, C input configuration This 2-bit field defines the Boolean evaluation associated with the selected input C in product term 0. 00 Force the C input in this product term to a logical zero 01 Pass the C input in this product term 10 Complement the C input in this product term 11 Force the C input in this product term to a logical one
9–8 PT0_DC	Product term 0, D input configuration This 2-bit field defines the Boolean evaluation associated with the selected input D in product term 0. 00 Force the D input in this product term to a logical zero 01 Pass the D input in this product term 10 Complement the D input in this product term 11 Force the D input in this product term to a logical one
7–6 PT1_AC	Product term 1, A input configuration This 2-bit field defines the Boolean evaluation associated with the selected input A in product term 1. 00 Force the A input in this product term to a logical zero 01 Pass the A input in this product term

Table continues on the next page...

AOI_BFCRT01*n* field descriptions (continued)

Field	Description
	10 Complement the A input in this product term 11 Force the A input in this product term to a logical one
5–4 PT1_BC	Product term 1, B input configuration This 2-bit field defines the Boolean evaluation associated with the selected input B in product term 1. 00 Force the B input in this product term to a logical zero 01 Pass the B input in this product term 10 Complement the B input in this product term 11 Force the B input in this product term to a logical one
3–2 PT1_CC	Product term 1, C input configuration This 2-bit field defines the Boolean evaluation associated with the selected input C in product term 1. 00 Force the C input in this product term to a logical zero 01 Pass the C input in this product term 10 Complement the C input in this product term 11 Force the C input in this product term to a logical one
PT1_DC	Product term 1, D input configuration This 2-bit field defines the Boolean evaluation associated with the selected input D in product term 1. 00 Force the D input in this product term to a logical zero 01 Pass the D input in this product term 10 Complement the D input in this product term 11 Force the D input in this product term to a logical one

15.3.2 Boolean Function Term 2 and 3 Configuration Register for EVENTn (AOI_BFCRT23*n*)

Address: E380h base + 1h offset + (2d × i), where i=0d to 3d

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read Write	PT2_AC		PT2_BC		PT2_CC		PT2_DC		PT3_AC		PT3_BC		PT3_CC		PT3_DC	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

AOI_BFCRT23*n* field descriptions

Field	Description
15–14 PT2_AC	Product term 2, A input configuration This 2-bit field defines the Boolean evaluation associated with the selected input A in product term 2. 00 Force the A input in this product term to a logical zero 01 Pass the A input in this product term 10 Complement the A input in this product term 11 Force the A input in this product term to a logical one

Table continues on the next page...

AOI_BFCRT23*n* field descriptions (continued)

Field	Description
13–12 PT2_BC	Product term 2, B input configuration This 2-bit field defines the Boolean evaluation associated with the selected input B in product term 2. 00 Force the B input in this product term to a logical zero 01 Pass the B input in this product term 10 Complement the B input in this product term 11 Force the B input in this product term to a logical one
11–10 PT2_CC	Product term 2, C input configuration This 2-bit field defines the Boolean evaluation associated with the selected input C in product term 2. 00 Force the C input in this product term to a logical zero 01 Pass the C input in this product term 10 Complement the C input in this product term 11 Force the C input in this product term to a logical one
9–8 PT2_DC	Product term 2, D input configuration This 2-bit field defines the Boolean evaluation associated with the selected input D in product term 2. 00 Force the D input in this product term to a logical zero 01 Pass the D input in this product term 10 Complement the D input in this product term 11 Force the D input in this product term to a logical one
7–6 PT3_AC	Product term 3, A input configuration This 2-bit field defines the Boolean evaluation associated with the selected input A in product term 3. 00 Force the A input in this product term to a logical zero 01 Pass the A input in this product term 10 Complement the A input in this product term 11 Force the A input in this product term to a logical one
5–4 PT3_BC	Product term 3, B input configuration This 2-bit field defines the Boolean evaluation associated with the selected input B in product term 3. 00 Force the B input in this product term to a logical zero 01 Pass the B input in this product term 10 Complement the B input in this product term 11 Force the B input in this product term to a logical one
3–2 PT3_CC	Product term 3, C input configuration This 2-bit field defines the Boolean evaluation associated with the selected input C in product term 3. 00 Force the C input in this product term to a logical zero 01 Pass the C input in this product term 10 Complement the C input in this product term 11 Force the C input in this product term to a logical one
PT3_DC	Product term 3, D input configuration This 2-bit field defines the Boolean evaluation associated with the selected input D in product term 3.

Table continues on the next page...

AOI_BFCRT23*n* field descriptions (continued)

Field	Description
	00 Force the D input in this product term to a logical zero 01 Pass the D input in this product term 10 Complement the D input in this product term 11 Force the D input in this product term to a logical one

15.4 Functional Description

The AOI is a highly programmable module for creating combinational boolean outputs for use as hardware triggers. Each AOI output channel, as shown in Figure 15-1, has one logic function:

- Evaluation of a combinational boolean expression as a sum of four products where each product term includes all four selected input sources available as true or complement values

A typical application of the AOI module is to be integrated with one or more inter-peripheral crossbar switch modules as illustrated in the following figure. The 20 external inputs are shared by two crossbar switch modules. The crossbar switch on the top is used to select the inputs to four 4-input AOI functions in the AOI module. The outputs of these four AOI functions are output from the AOI module and are added to the original 20 external inputs to provide a total of 24 inputs to the bottom crossbar switch. As a result, the bottom crossbar can not only direct any of the original 20 external inputs to any of its outputs, it can also now direct any one of four 4-input AOI functions of those external inputs to any of its outputs.

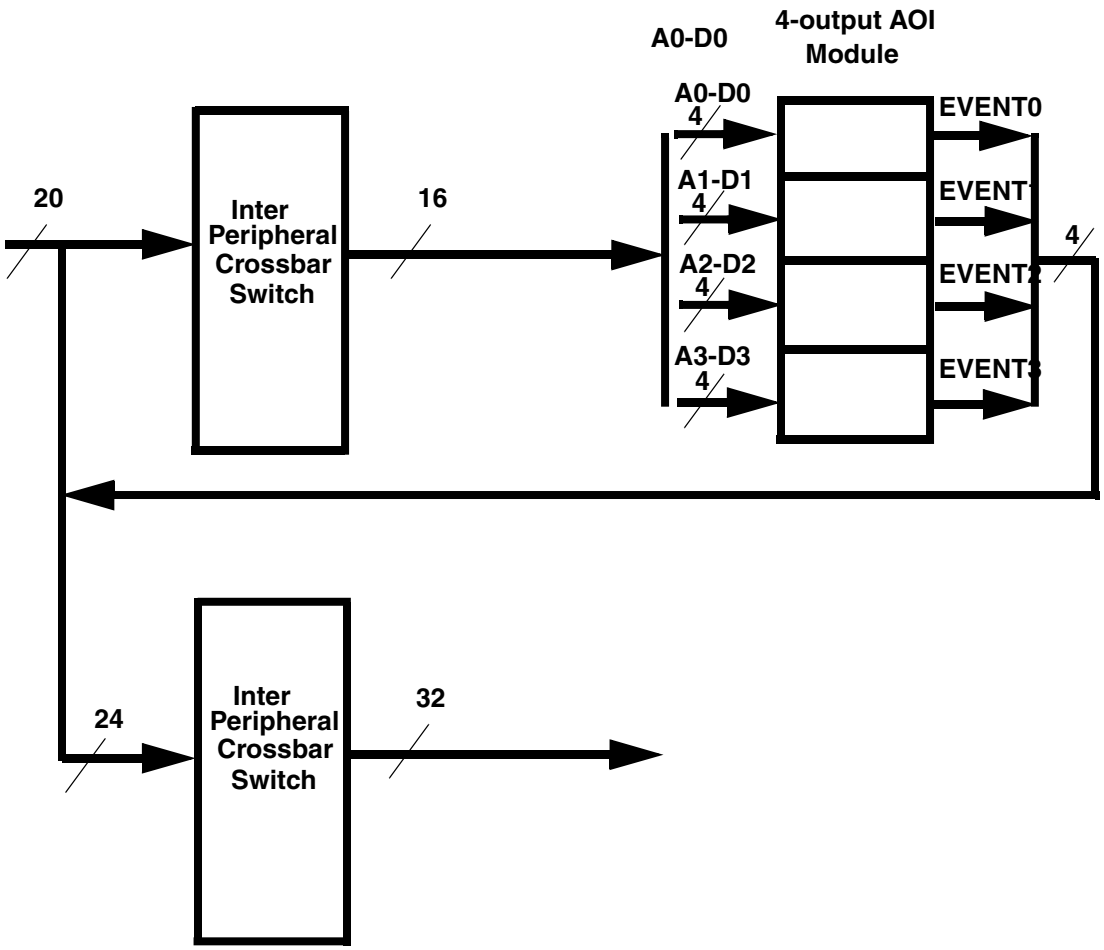


Figure 15-2. Integration Example of AOI with two Inter-Peripheral Crossbar Switches

15.4.1 Configuration Examples for the Boolean Function Evaluation

This section presents examples of the programming model configuration for simple boolean expressions.

The AOI module provides a universal boolean function generator using a four-term sum of products expression with each product term containing true or complement values of the four selected event inputs (A, B, C, D). Specifically, the event output is defined by the following “4 x 4” boolean expression:

```
EVENTn
    = (0,An,~An,1) & (0,Bn,~Bn,1) & (0,Cn,~Cn,1) & (0,Dn,~Dn,1)// product term 0
    | (0,An,~An,1) & (0,Bn,~Bn,1) & (0,Cn,~Cn,1) & (0,Dn,~Dn,1)// product term 1
```

```
    | (0,An,~An,1) & (0,Bn,~Bn,1) & (0,Cn,~Cn,1) & (0,Dn,~Dn,1)// product term 2
    | (0,An,~An,1) & (0,Bn,~Bn,1) & (0,Cn,~Cn,1) & (0,Dn,~Dn,1)// product term 3
```

where each selected input term in each product term can be configured to produce a logical 0 or 1 or pass the true or complement of the selected event input. Each product term uses eight bits of configuration information, two bits for each of the four selected event inputs. The actual boolean expression implemented in each channel is:

```
EVENTn
    = (PT0_AC[0] & A |  PT0_AC[1] & ~A)// product term 0
                     & (PT0_BC[0] &  B | PTO_BC[1] & ~B)
                     & (PT0_CC[0] &  C | PTO_CC[1] & ~C)
                     & (PT0_DC[0] &  D | PTO_DC[1] & ~D)

    | (PT1_AC[0] & A |  PT1_AC[1] & ~A)// product term 1
                     & (PT1_BC[0] &  B | PT1_BC[1] & ~B)
                     & (PT1_CC[0] &  C | PT1_CC[1] & ~C)
                     & (PT1_DC[0] &  D | PT1_DC[1] & ~D)

    | (PT2_AC[0] & A |  PT2_AC[1] & ~A)// product term 2
                     & (PT2_BC[0] &  B | PT2_BC[1] & ~B)
                     & (PT2_CC[0] &  C | PT2_CC[1] & ~C)
                     & (PT2_DC[0] &  D | PT2_DC[1] & ~D)

    | (PT3_AC[0] & A |  PT3_AC[1] & ~A)// product term 3
                     & (PT3_BC[0] &  B | PT3_BC[1] & ~B)
                     & (PT3_CC[0] &  C | PT3_CC[1] & ~C)
                     & (PT3_DC[0] &  D | PT3_DC[1] & ~D)
```

where the bits of the combined {BFECRT01n,BFCRT23n} registers correspond to the `PT{0-3}_{A,B,C,D}C[1:0]` terms in the equation.

Consider the settings of the combined 32-bit {BFECRT01n,BFCRT23n} registers for several simple boolean expressions as shown in Table 15-1.

Table 15-1. IEVENT_BFECRn Values for Simple Boolean Expressions

Event Output Expression	PT0	PT1	PT2	PT3	{BFCRT01, BFCRT23}
A & B	A & B	0	0	0	01011111_00000000_00000000_00000000
A & B & C	A & B & C	0	0	0	01010111_00000000_00000000_00000000
(A & B & C) + D	A & B & C	D	0	0	01010111_11111101_00000000_00000000
A + B + C + D	A	B	C	D	01111111_11011111_11110111_11111101
(A & ~B) + (~A & B)	A & ~B	~A & B	0	0	01101111_10011111_00000000_00000000

As can be seen in these examples, the resulting logic provides a simple yet powerful boolean function evaluation for defining an event output.

15.4.2 AOI Timing Between Inputs and Outputs

Each EVENTn output of the AOI module is a combination function of its four dedicated inputs An, Bn, Cn, and Dn. Propagation through the AOI and any associated inter-peripheral crossbar switch modules is intended to be single bus clock cycle.

Chapter 16
Inter-Peripheral Crossbar Switch A (XBARA)

16.1 Introduction

NOTE

For the chip-specific implementation details of this module's instances, see the chip configuration information.

16.1.1 Overview

This module implements an array of M N-input combinational muxes. All muxes share the same N inputs in the same order, but each mux has its own independent select field.

The intended application of this module is to provide a flexible crossbar switch function that allows any input (typically from external GPIO or internal module outputs) to be connected to any output (typically to external GPIO or internal module inputs) under user control. This is used to allow user configuration of data paths between internal modules and between internal modules and GPIO.

A subset of the muxes can be configured to support edge detection and either interrupt or DMA request generation based on detected signal edges on the mux output. This allows signal transitions on the signals feeding the crossbar to trigger interrupts or initiate data transfers via DMA into or out of other system modules.

16.1.2 Features

The XBAR module design includes these distinctive features:

- M identical N-input muxes with individual select fields.
- Edge detection with associated interrupt or DMA request generation for a subset of mux outputs.

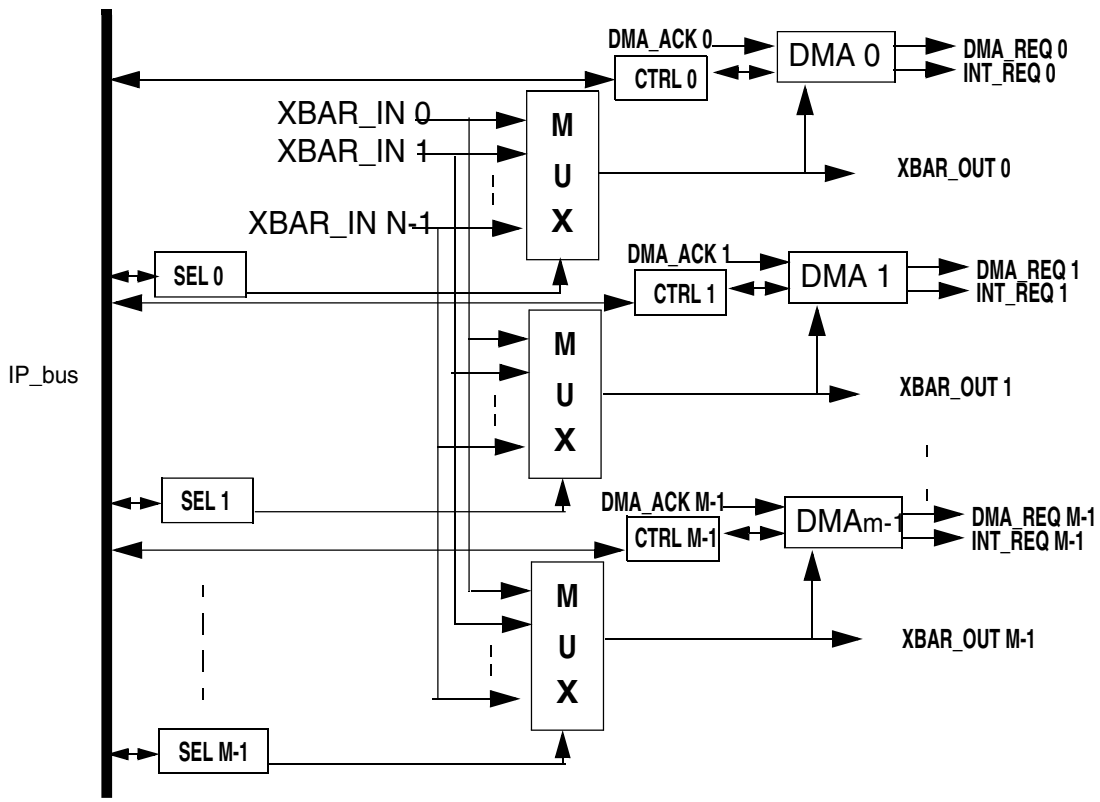
- Memory mapped registers with IPBus interface for select and control fields.
- Register write protection input signal.

16.1.3 Modes of Operation

The XBAR module design operates in only a single mode of operation: Functional Mode. The various counting modes are detailed in Functional Mode.

16.1.4 Block Diagram

The block diagram for XBAR is shown in Figure 16-1.



Note: Control field support can be restricted to a subset of outputs. CTRLn represents the memory mapped control fields in the XBAR_CTRL registers for each corresponding XBAR_OUT output. Control fields provide edge status, edge selection, and interrupt/DMA generation controls.

Figure 16-1. XBAR Block Diagram

16.2 Signal Descriptions

The following table summarizes the module's external signals.

Table 16-1. Control Signal Properties

Name	I/O Type	Function	Reset State	Notes
XBAR_OUT [0:NUMOUT-1]	O	Mux Outputs with configurable width	*	
XBAR_IN [0:NUMIN-1]	I	Mux Inputs with configurable width	*	
DMA_REQ	O	DMA request	0	
INT_REQ	O	Interrupt request	0	
DMA_ACK	I	DMA acknowledge	0	

At reset, each output XBAR_OUT[*] contains the reset value of the signal driving XBAR_IN[0].

16.2.1 XBAR_OUT[0:NUM_OUT-1] - MUX Outputs

This is a one-dimensional array of the mux outputs. The value on each output XBAR_OUT[n] is determined by the setting of the corresponding memory mapped register SELn such that XBAR_OUT[n] = XBAR_IN[SELn].

16.2.2 XBAR_IN[0:NUM_IN-1] - MUX Inputs

This is a one-dimensional array consisting of the inputs shared by each mux. All muxes share the same inputs in the same order.

16.2.3 DMA_REQ[n] - DMA Request Output(s)

DMA_REQ[n] is a DMA request to the DMA controller.

16.2.4 DMA_ACK[n] - DMA Acknowledge Input(s)

DMA_ACK[n] is a DMA acknowledge input from the DMA controller.

16.2.5 INT_REQ[n] - Interrupt Request Output(s)

INT_REQ[n] is an interrupt request output to the interrupt controller.

16.3 Memory Map and Register Descriptions

This section provides information about the XBARA instance of the inter-peripheral crossbar switch. Refer to XBARB register details for information about that instance's registers.

The XBAR module has select registers and control registers.

In the XBAR select registers, the SELn fields select which of the shared inputs (XBAR_IN[*]) is muxed to each mux output (XBAR_OUT[*]). There is one SELn field per mux and therefore one per XBAR_OUT output. Crossbar output XBAR_OUT[n] presents the value of XBAR_IN[SELn]. Each select register contains two SELn fields. In the first select register, the LSBs contain the select field for mux 0, and the MSBs contain the select field for mux 1. The pattern repeats in subsequent select registers.

The actual signals connected to XBAR_IN and XBAR_OUT are application specific and are described in the Chip Configuration details.

The XBAR control registers configure edge detection, interrupt, and DMA features for a subset of the XBAR_OUT[*] outputs.

XBARA memory map

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
E340	Crossbar A Select Register 0 (XBARA_SEL0)	16	R/W	0000h	16.3.1/297
E341	Crossbar A Select Register 1 (XBARA_SEL1)	16	R/W	0000h	16.3.2/298
E342	Crossbar A Select Register 2 (XBARA_SEL2)	16	R/W	0000h	16.3.3/298
E343	Crossbar A Select Register 3 (XBARA_SEL3)	16	R/W	0000h	16.3.4/299
E344	Crossbar A Select Register 4 (XBARA_SEL4)	16	R/W	0000h	16.3.5/299
E345	Crossbar A Select Register 5 (XBARA_SEL5)	16	R/W	0000h	16.3.6/300
E346	Crossbar A Select Register 6 (XBARA_SEL6)	16	R/W	0000h	16.3.7/300
E347	Crossbar A Select Register 7 (XBARA_SEL7)	16	R/W	0000h	16.3.8/301
E348	Crossbar A Select Register 8 (XBARA_SEL8)	16	R/W	0000h	16.3.9/301
E349	Crossbar A Select Register 9 (XBARA_SEL9)	16	R/W	0000h	16.3.10/302

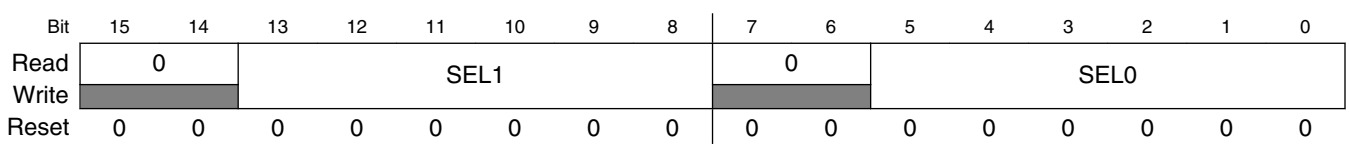
Table continues on the next page...

XBARA memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
E34A	Crossbar A Select Register 10 (XBARA_SEL10)	16	R/W	0000h	16.3.11/302
E34B	Crossbar A Select Register 11 (XBARA_SEL11)	16	R/W	0000h	16.3.12/303
E34C	Crossbar A Select Register 12 (XBARA_SEL12)	16	R/W	0000h	16.3.13/303
E34D	Crossbar A Select Register 13 (XBARA_SEL13)	16	R/W	0000h	16.3.14/304
E34E	Crossbar A Select Register 14 (XBARA_SEL14)	16	R/W	0000h	16.3.15/304
E34F	Crossbar A Select Register 15 (XBARA_SEL15)	16	R/W	0000h	16.3.16/305
E350	Crossbar A Select Register 16 (XBARA_SEL16)	16	R/W	0000h	16.3.17/305
E351	Crossbar A Select Register 17 (XBARA_SEL17)	16	R/W	0000h	16.3.18/306
E352	Crossbar A Select Register 18 (XBARA_SEL18)	16	R/W	0000h	16.3.19/306
E353	Crossbar A Select Register 19 (XBARA_SEL19)	16	R/W	0000h	16.3.20/307
E354	Crossbar A Select Register 20 (XBARA_SEL20)	16	R/W	0000h	16.3.21/307
E355	Crossbar A Select Register 21 (XBARA_SEL21)	16	R/W	0000h	16.3.22/308
E356	Crossbar A Select Register 22 (XBARA_SEL22)	16	R/W	0000h	16.3.23/308
E357	Crossbar A Select Register 23 (XBARA_SEL23)	16	R/W	0000h	16.3.24/309
E358	Crossbar A Select Register 24 (XBARA_SEL24)	16	R/W	0000h	16.3.25/309
E359	Crossbar A Select Register 25 (XBARA_SEL25)	16	R/W	0000h	16.3.26/310
E35A	Crossbar A Select Register 26 (XBARA_SEL26)	16	R/W	0000h	16.3.27/310
E35B	Crossbar A Select Register 27 (XBARA_SEL27)	16	R/W	0000h	16.3.28/311
E35C	Crossbar A Select Register 28 (XBARA_SEL28)	16	R/W	0000h	16.3.29/311
E35D	Crossbar A Select Register 29 (XBARA_SEL29)	16	R/W	0000h	16.3.30/312
E35E	Crossbar A Control Register 0 (XBARA_CTRL0)	16	R/W	0000h	16.3.31/312
E35F	Crossbar A Control Register 1 (XBARA_CTRL1)	16	R/W	0000h	16.3.32/314

16.3.1 Crossbar A Select Register 0 (XBARA_SEL0)

Address: E340h base + 0h offset = E340h



XBARA_SEL0 field descriptions

Field	Description
15–14 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
13–8 SEL1	Input (XBARA_INn) to be muxed to XBARA_OUT1 (refer to Functional Description section for input/output assignment)

Table continues on the next page...

XBARA_SEL0 field descriptions (continued)

Field	Description
7–6 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
SEL0	Input (XBARA_INn) to be muxed to XBARA_OUT0 (refer to Functional Description section for input/output assignment)

16.3.2 Crossbar A Select Register 1 (XBARA_SEL1)

Address: E340h base + 1h offset = E341h

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	0		SEL3						0		SEL2					
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

XBARA_SEL1 field descriptions

Field	Description
15–14 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
13–8 SEL3	Input (XBARA_INn) to be muxed to XBARA_OUT3 (refer to Functional Description section for input/output assignment)
7–6 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
SEL2	Input (XBARA_INn) to be muxed to XBARA_OUT2 (refer to Functional Description section for input/output assignment)

16.3.3 Crossbar A Select Register 2 (XBARA_SEL2)

Address: E340h base + 2h offset = E342h

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	0		SEL5						0		SEL4					
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

XBARA_SEL2 field descriptions

Field	Description
15–14 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
13–8 SEL5	Input (XBARA_INn) to be muxed to XBARA_OUT5 (refer to Functional Description section for input/output assignment)
7–6 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

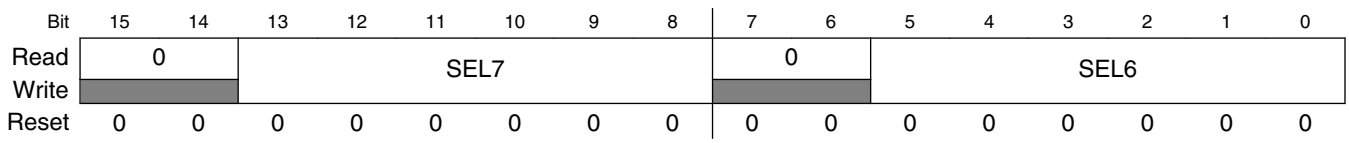
Table continues on the next page...

XBARA_SEL2 field descriptions (continued)

Field	Description
SEL4	Input (XBARA_INn) to be muxed to XBARA_OUT4 (refer to Functional Description section for input/output assignment)

16.3.4 Crossbar A Select Register 3 (XBARA_SEL3)

Address: E340h base + 3h offset = E343h

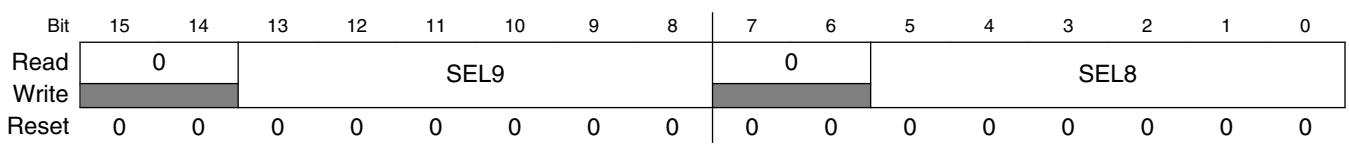


XBARA_SEL3 field descriptions

Field	Description
15–14 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
13–8 SEL7	Input (XBARA_INn) to be muxed to XBARA_OUT7 (refer to Functional Description section for input/output assignment)
7–6 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
SEL6	Input (XBARA_INn) to be muxed to XBARA_OUT6 (refer to Functional Description section for input/output assignment)

16.3.5 Crossbar A Select Register 4 (XBARA_SEL4)

Address: E340h base + 4h offset = E344h



XBARA_SEL4 field descriptions

Field	Description
15–14 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
13–8 SEL9	Input (XBARA_INn) to be muxed to XBARA_OUT9 (refer to Functional Description section for input/output assignment)
7–6 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
SEL8	Input (XBARA_INn) to be muxed to XBARA_OUT8 (refer to Functional Description section for input/output assignment)

16.3.6 Crossbar A Select Register 5 (XBARA_SEL5)

Address: E340h base + 5h offset = E345h

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	0		SEL11						0		SEL10					
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

XBARA_SEL5 field descriptions

Field	Description
15–14 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
13–8 SEL11	Input (XBARA_INn) to be muxed to XBARA_OUT11 (refer to Functional Description section for input/output assignment)
7–6 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
SEL10	Input (XBARA_INn) to be muxed to XBARA_OUT10 (refer to Functional Description section for input/output assignment)

16.3.7 Crossbar A Select Register 6 (XBARA_SEL6)

Address: E340h base + 6h offset = E346h

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	0		SEL13						0		SEL12					
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

XBARA_SEL6 field descriptions

Field	Description
15–14 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
13–8 SEL13	Input (XBARA_INn) to be muxed to XBARA_OUT13 (refer to Functional Description section for input/output assignment)
7–6 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
SEL12	Input (XBARA_INn) to be muxed to XBARA_OUT12 (refer to Functional Description section for input/output assignment)

16.3.8 Crossbar A Select Register 7 (XBARA_SEL7)

Address: E340h base + 7h offset = E347h

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	0		SEL15						0		SEL14					
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

XBARA_SEL7 field descriptions

Field	Description
15–14 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
13–8 SEL15	Input (XBARA_INn) to be muxed to XBARA_OUT15 (refer to Functional Description section for input/output assignment)
7–6 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
SEL14	Input (XBARA_INn) to be muxed to XBARA_OUT14 (refer to Functional Description section for input/output assignment)

16.3.9 Crossbar A Select Register 8 (XBARA_SEL8)

Address: E340h base + 8h offset = E348h

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	0		SEL17						0		SEL16					
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

XBARA_SEL8 field descriptions

Field	Description
15–14 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
13–8 SEL17	Input (XBARA_INn) to be muxed to XBARA_OUT17 (refer to Functional Description section for input/output assignment)
7–6 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
SEL16	Input (XBARA_INn) to be muxed to XBARA_OUT16 (refer to Functional Description section for input/output assignment)

16.3.10 Crossbar A Select Register 9 (XBARA_SEL9)

Address: E340h base + 9h offset = E349h

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	0		SEL19						0		SEL18					
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

XBARA_SEL9 field descriptions

Field	Description
15–14 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
13–8 SEL19	Input (XBARA_INn) to be muxed to XBARA_OUT19 (refer to Functional Description section for input/output assignment)
7–6 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
SEL18	Input (XBARA_INn) to be muxed to XBARA_OUT18 (refer to Functional Description section for input/output assignment)

16.3.11 Crossbar A Select Register 10 (XBARA_SEL10)

Address: E340h base + Ah offset = E34Ah

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	0		SEL21						0		SEL20					
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

XBARA_SEL10 field descriptions

Field	Description
15–14 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
13–8 SEL21	Input (XBARA_INn) to be muxed to XBARA_OUT21 (refer to Functional Description section for input/output assignment)
7–6 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
SEL20	Input (XBARA_INn) to be muxed to XBARA_OUT20 (refer to Functional Description section for input/output assignment)

16.3.12 Crossbar A Select Register 11 (XBARA_SEL11)

Address: E340h base + Bh offset = E34Bh

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	0		SEL23						0		SEL22					
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

XBARA_SEL11 field descriptions

Field	Description
15–14 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
13–8 SEL23	Input (XBARA_INn) to be muxed to XBARA_OUT23 (refer to Functional Description section for input/output assignment)
7–6 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
SEL22	Input (XBARA_INn) to be muxed to XBARA_OUT22 (refer to Functional Description section for input/output assignment)

16.3.13 Crossbar A Select Register 12 (XBARA_SEL12)

Address: E340h base + Ch offset = E34Ch

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	0		SEL25						0		SEL24					
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

XBARA_SEL12 field descriptions

Field	Description
15–14 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
13–8 SEL25	Input (XBARA_INn) to be muxed to XBARA_OUT25 (refer to Functional Description section for input/output assignment)
7–6 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
SEL24	Input (XBARA_INn) to be muxed to XBARA_OUT24 (refer to Functional Description section for input/output assignment)

16.3.14 Crossbar A Select Register 13 (XBARA_SEL13)

Address: E340h base + Dh offset = E34Dh

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	0		SEL27						0		SEL26					
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

XBARA_SEL13 field descriptions

Field	Description
15–14 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
13–8 SEL27	Input (XBARA_INn) to be muxed to XBARA_OUT27 (refer to Functional Description section for input/output assignment)
7–6 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
SEL26	Input (XBARA_INn) to be muxed to XBARA_OUT26 (refer to Functional Description section for input/output assignment)

16.3.15 Crossbar A Select Register 14 (XBARA_SEL14)

Address: E340h base + Eh offset = E34Eh

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	0		SEL29						0		SEL28					
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

XBARA_SEL14 field descriptions

Field	Description
15–14 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
13–8 SEL29	Input (XBARA_INn) to be muxed to XBARA_OUT29 (refer to Functional Description section for input/output assignment)
7–6 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
SEL28	Input (XBARA_INn) to be muxed to XBARA_OUT28 (refer to Functional Description section for input/output assignment)

16.3.16 Crossbar A Select Register 15 (XBARA_SEL15)

Address: E340h base + Fh offset = E34Fh

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	0		SEL31						0		SEL30					
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

XBARA_SEL15 field descriptions

Field	Description
15–14 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
13–8 SEL31	Input (XBARA_INn) to be muxed to XBARA_OUT31 (refer to Functional Description section for input/output assignment)
7–6 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
SEL30	Input (XBARA_INn) to be muxed to XBARA_OUT30 (refer to Functional Description section for input/output assignment)

16.3.17 Crossbar A Select Register 16 (XBARA_SEL16)

Address: E340h base + 10h offset = E350h

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	0		SEL33						0		SEL32					
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

XBARA_SEL16 field descriptions

Field	Description
15–14 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
13–8 SEL33	Input (XBARA_INn) to be muxed to XBARA_OUT33 (refer to Functional Description section for input/output assignment)
7–6 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
SEL32	Input (XBARA_INn) to be muxed to XBARA_OUT32 (refer to Functional Description section for input/output assignment)

16.3.18 Crossbar A Select Register 17 (XBARA_SEL17)

Address: E340h base + 11h offset = E351h

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	0		SEL35						0		SEL34					
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

XBARA_SEL17 field descriptions

Field	Description
15–14 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
13–8 SEL35	Input (XBARA_INn) to be muxed to XBARA_OUT35 (refer to Functional Description section for input/output assignment)
7–6 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
SEL34	Input (XBARA_INn) to be muxed to XBARA_OUT34 (refer to Functional Description section for input/output assignment)

16.3.19 Crossbar A Select Register 18 (XBARA_SEL18)

Address: E340h base + 12h offset = E352h

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	0		SEL37						0		SEL36					
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

XBARA_SEL18 field descriptions

Field	Description
15–14 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
13–8 SEL37	Input (XBARA_INn) to be muxed to XBARA_OUT37 (refer to Functional Description section for input/output assignment)
7–6 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
SEL36	Input (XBARA_INn) to be muxed to XBARA_OUT36 (refer to Functional Description section for input/output assignment)

16.3.20 Crossbar A Select Register 19 (XBARA_SEL19)

Address: E340h base + 13h offset = E353h

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	0		SEL39						0		SEL38					
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

XBARA_SEL19 field descriptions

Field	Description
15–14 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
13–8 SEL39	Input (XBARA_INn) to be muxed to XBARA_OUT39 (refer to Functional Description section for input/output assignment)
7–6 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
SEL38	Input (XBARA_INn) to be muxed to XBARA_OUT38 (refer to Functional Description section for input/output assignment)

16.3.21 Crossbar A Select Register 20 (XBARA_SEL20)

Address: E340h base + 14h offset = E354h

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	0		SEL41						0		SEL40					
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

XBARA_SEL20 field descriptions

Field	Description
15–14 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
13–8 SEL41	Input (XBARA_INn) to be muxed to XBARA_OUT41 (refer to Functional Description section for input/output assignment)
7–6 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
SEL40	Input (XBARA_INn) to be muxed to XBARA_OUT40 (refer to Functional Description section for input/output assignment)

16.3.22 Crossbar A Select Register 21 (XBARA_SEL21)

Address: E340h base + 15h offset = E355h

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	0		SEL43						0		SEL42					
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

XBARA_SEL21 field descriptions

Field	Description
15–14 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
13–8 SEL43	Input (XBARA_INn) to be muxed to XBARA_OUT43 (refer to Functional Description section for input/output assignment)
7–6 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
SEL42	Input (XBARA_INn) to be muxed to XBARA_OUT42 (refer to Functional Description section for input/output assignment)

16.3.23 Crossbar A Select Register 22 (XBARA_SEL22)

Address: E340h base + 16h offset = E356h

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	0		SEL45						0		SEL44					
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

XBARA_SEL22 field descriptions

Field	Description
15–14 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
13–8 SEL45	Input (XBARA_INn) to be muxed to XBARA_OUT45 (refer to Functional Description section for input/output assignment)
7–6 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
SEL44	Input (XBARA_INn) to be muxed to XBARA_OUT44 (refer to Functional Description section for input/output assignment)

16.3.24 Crossbar A Select Register 23 (XBARA_SEL23)

Address: E340h base + 17h offset = E357h

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	0		SEL47						0		SEL46					
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

XBARA_SEL23 field descriptions

Field	Description
15–14 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
13–8 SEL47	Input (XBARA_INn) to be muxed to XBARA_OUT47 (refer to Functional Description section for input/output assignment)
7–6 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
SEL46	Input (XBARA_INn) to be muxed to XBARA_OUT46 (refer to Functional Description section for input/output assignment)

16.3.25 Crossbar A Select Register 24 (XBARA_SEL24)

Address: E340h base + 18h offset = E358h

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	0		SEL49						0		SEL48					
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

XBARA_SEL24 field descriptions

Field	Description
15–14 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
13–8 SEL49	Input (XBARA_INn) to be muxed to XBARA_OUT49 (refer to Functional Description section for input/output assignment)
7–6 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
SEL48	Input (XBARA_INn) to be muxed to XBARA_OUT48 (refer to Functional Description section for input/output assignment)

16.3.26 Crossbar A Select Register 25 (XBARA_SEL25)

Address: E340h base + 19h offset = E359h

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	0		SEL51						0		SEL50					
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

XBARA_SEL25 field descriptions

Field	Description
15–14 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
13–8 SEL51	Input (XBARA_INn) to be muxed to XBARA_OUT51 (refer to Functional Description section for input/output assignment)
7–6 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
SEL50	Input (XBARA_INn) to be muxed to XBARA_OUT50 (refer to Functional Description section for input/output assignment)

16.3.27 Crossbar A Select Register 26 (XBARA_SEL26)

Address: E340h base + 1Ah offset = E35Ah

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	0		SEL53						0		SEL52					
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

XBARA_SEL26 field descriptions

Field	Description
15–14 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
13–8 SEL53	Input (XBARA_INn) to be muxed to XBARA_OUT53 (refer to Functional Description section for input/output assignment)
7–6 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
SEL52	Input (XBARA_INn) to be muxed to XBARA_OUT52 (refer to Functional Description section for input/output assignment)

16.3.28 Crossbar A Select Register 27 (XBARA_SEL27)

Address: E340h base + 1Bh offset = E35Bh

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	0		SEL55						0		SEL54					
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

XBARA_SEL27 field descriptions

Field	Description
15–14 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
13–8 SEL55	Input (XBARA_INn) to be muxed to XBARA_OUT55 (refer to Functional Description section for input/output assignment)
7–6 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
SEL54	Input (XBARA_INn) to be muxed to XBARA_OUT54 (refer to Functional Description section for input/output assignment)

16.3.29 Crossbar A Select Register 28 (XBARA_SEL28)

Address: E340h base + 1Ch offset = E35Ch

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	0		SEL57						0		SEL56					
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

XBARA_SEL28 field descriptions

Field	Description
15–14 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
13–8 SEL57	Input (XBARA_INn) to be muxed to XBARA_OUT57 (refer to Functional Description section for input/output assignment)
7–6 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
SEL56	Input (XBARA_INn) to be muxed to XBARA_OUT56 (refer to Functional Description section for input/output assignment)

16.3.30 Crossbar A Select Register 29 (XBARA_SEL29)

Address: E340h base + 1Dh offset = E35Dh

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	0		0						0		SEL58					
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

XBARA_SEL29 field descriptions

Field	Description
15–14 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
13–8 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
7–6 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
SEL58	Input (XBARA_INn) to be muxed to XBARA_OUT58 (refer to Functional Description section for input/ output assignment)

16.3.31 Crossbar A Control Register 0 (XBARA_CTRL0)

Use this register to configure edge detection, interrupt, and DMA features for the XBAR_OUT0 and XBAR_OUT1 outputs.

The XBAR_CTRL registers are organized similarly to the XBAR_SEL registers, with control fields for two XBAR_OUT outputs in each register. In control register 0, the LSBs contain the control fields for XBAR_OUT0, and the MSBs contain the control fields for XBAR_OUT1.

Address: E340h base + 1Eh offset = E35Eh

Bit	15	14	13	12	11	10	9	8
Read	0			STS1	EDGE1		IEN1	DEN1
Write				w1c				
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
Read	0			STS0	EDGE0		IEN0	DEN0
Write				w1c				
Reset	0	0	0	0	0	0	0	0

XBARA_CTRL0 field descriptions

Field	Description
15–13 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
12 STS1	Edge detection status for XBAR_OUT1 This bit reflects the results of edge detection for XBAR_OUT1. This field is set to 1 when an edge consistent with the current setting of EDGE1 is detected on XBAR_OUT1. This field is cleared by writing 1 to it or by a DMA_ACK1 reception when DEN1 is set. Writing 0 to the field has no effect. When interrupt or DMA functionality is enabled for XBAR_OUT1, this field is 1 when the interrupt or DMA request is asserted and 0 when the interrupt or DMA request has been cleared. 0 Active edge not yet detected on XBAR_OUT1 1 Active edge detected on XBAR_OUT1
11–10 EDGE1	Active edge for edge detection on XBAR_OUT1 This field selects which edges on XBAR_OUT1 cause STS1 to assert. 00 STS1 never asserts 01 STS1 asserts on rising edges of XBAR_OUT1 10 STS1 asserts on falling edges of XBAR_OUT1 11 STS1 asserts on rising and falling edges of XBAR_OUT1
9 IEN1	Interrupt Enable for XBAR_OUT1 This bit enables the interrupt function on the corresponding XBAR_OUT1 output. When the interrupt is enabled, the output INT_REQ1 reflects the value STS1. When the interrupt is disabled, INT_REQ1 remains low. The interrupt request is cleared by writing a 1 to STS1. **Restriction:** IEN1 and DEN1 should not both be set to 1. 0 Interrupt disabled 1 Interrupt enabled
8 DEN1	DMA Enable for XBAR_OUT1 This bit enables the DMA function on the corresponding XBAR_OUT1 output. When enabled, DMA_REQ1 presents the value STS1. When disabled, the DMA_REQ1 output remains low. **Restriction:** IEN1 and DEN1 should not both be set to 1. 0 DMA disabled 1 DMA enabled
7–5 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
4 STS0	Edge detection status for XBAR_OUT0 This bit reflects the results of edge detection for XBAR_OUT0. This field is set to 1 when an edge consistent with the current setting of EDGE0 is detected on XBAR_OUT0. This field is cleared by writing 1 to it or by a DMA_ACK0 reception when DEN0 is set. Writing 0 to the field has no effect. When interrupt or DMA functionality is enabled for XBAR_OUT0, this field is 1 when the interrupt or DMA request is asserted and 0 when the interrupt or DMA request has been cleared.

Table continues on the next page...

XBARA_CTRL0 field descriptions (continued)

Field	Description
	0 Active edge not yet detected on XBAR_OUT0 1 Active edge detected on XBAR_OUT0
3–2 EDGE0	Active edge for edge detection on XBAR_OUT0 This field selects which edges on XBAR_OUT0 cause STS0 to assert. 00 STS0 never asserts 01 STS0 asserts on rising edges of XBAR_OUT0 10 STS0 asserts on falling edges of XBAR_OUT0 11 STS0 asserts on rising and falling edges of XBAR_OUT0
1 IEN0	Interrupt Enable for XBAR_OUT0 This bit enables the interrupt function on the corresponding XBAR_OUT0 output. When the interrupt is enabled, the output INT_REQ0 reflects the value STS0. When the interrupt is disabled, INT_REQ0 remains low. The interrupt request is cleared by writing a 1 to STS0. **Restriction:** IEN0 and DEN0 should not both be set to 1. 0 Interrupt disabled 1 Interrupt enabled
0 DEN0	DMA Enable for XBAR_OUT0 This bit enables the DMA function on the corresponding XBAR_OUT0 output. When enabled, DMA_REQ0 presents the value STS0. When disabled, the DMA_REQ0 output remains low. **Restriction:** IEN0 and DEN0 should not both be set to 1. 0 DMA disabled 1 DMA enabled

16.3.32 Crossbar A Control Register 1 (XBARA_CTRL1)

Use this register to configure edge detection, interrupt, and DMA features for the XBAR_OUT2 and XBAR_OUT3 outputs.

The XBAR_CTRL registers are organized similarly to the XBAR_SEL registers, with control fields for two XBAR_OUT outputs in each register. In control register 1, the LSBs contain the control fields for XBAR_OUT2, and the MSBs contain the control fields for XBAR_OUT3.

Address: E340h base + 1Fh offset = E35Fh

Bit	15	14	13	12	11	10	9	8
Read	0			STS3	EDGE3		IEN3	DEN3
Write				w1c				
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
Read	0			STS2	EDGE2		IEN2	DEN2
Write				w1c				
Reset	0	0	0	0	0	0	0	0

XBARA_CTRL1 field descriptions

Field	Description
15–13 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
12 STS3	Edge detection status for XBAR_OUT3 This bit reflects the results of edge detection for XBAR_OUT3. This field is set to 1 when an edge consistent with the current setting of EDGE3 is detected on XBAR_OUT3. This field is cleared by writing 1 to it or by a DMA_ACK3 reception when DEN3 is set. Writing 0 to the field has no effect. When interrupt or DMA functionality is enabled for XBAR_OUT3, this field is 1 when the interrupt or DMA request is asserted and 0 when the interrupt or DMA request has been cleared. 0 Active edge not yet detected on XBAR_OUT3 1 Active edge detected on XBAR_OUT3
11–10 EDGE3	Active edge for edge detection on XBAR_OUT3 This field selects which edges on XBAR_OUT3 cause STS3 to assert. 00 STS3 never asserts 01 STS3 asserts on rising edges of XBAR_OUT3 10 STS3 asserts on falling edges of XBAR_OUT3 11 STS3 asserts on rising and falling edges of XBAR_OUT3
9 IEN3	Interrupt Enable for XBAR_OUT3 This bit enables the interrupt function on the corresponding XBAR_OUT3 output. When the interrupt is enabled, the output INT_REQ3 reflects the value STS3. When the interrupt is disabled, INT_REQ3 remains low. The interrupt request is cleared by writing a 1 to STS3. **Restriction:** IEN3 and DEN3 should not both be set to 1. 0 Interrupt disabled 1 Interrupt enabled
8 DEN3	DMA Enable for XBAR_OUT3 This bit enables the DMA function on the corresponding XBAR_OUT3 output. When enabled, DMA_REQ3 presents the value STS3. When disabled, the DMA_REQ3 output remains low. **Restriction:** IEN3 and DEN3 should not both be set to 1. 0 DMA disabled 1 DMA enabled
7–5 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
4 STS2	Edge detection status for XBAR_OUT2 This bit reflects the results of edge detection for XBAR_OUT2.

Table continues on the next page...

XBARA_CTRL1 field descriptions (continued)

Field	Description
	This field is set to 1 when an edge consistent with the current setting of EDGE2 is detected on XBAR_OUT2. This field is cleared by writing 1 to it or by a DMA_ACK2 reception when DEN2 is set. Writing 0 to the field has no effect. When interrupt or DMA functionality is enabled for XBAR_OUT2, this field is 1 when the interrupt or DMA request is asserted and 0 when the interrupt or DMA request has been cleared. 0 Active edge not yet detected on XBAR_OUT2 1 Active edge detected on XBAR_OUT2
3–2 EDGE2	Active edge for edge detection on XBAR_OUT2 This field selects which edges on XBAR_OUT2 cause STS2 to assert. 00 STS2 never asserts 01 STS2 asserts on rising edges of XBAR_OUT2 10 STS2 asserts on falling edges of XBAR_OUT2 11 STS2 asserts on rising and falling edges of XBAR_OUT2
1 IEN2	Interrupt Enable for XBAR_OUT2 This bit enables the interrupt function on the corresponding XBAR_OUT2 output. When the interrupt is enabled, the output INT_REQ2 reflects the value STS2. When the interrupt is disabled, INT_REQ2 remains low. The interrupt request is cleared by writing a 1 to STS2. **Restriction:** IEN2 and DEN2 should not both be set to 1. 0 Interrupt disabled 1 Interrupt enabled
0 DEN2	DMA Enable for XBAR_OUT2 This bit enables the DMA function on the corresponding XBAR_OUT2 output. When enabled, DMA_REQ2 presents the value STS2. When disabled, the DMA_REQ2 output remains low. **Restriction:** IEN2 and DEN2 should not both be set to 1. 0 DMA disabled 1 DMA enabled

16.4 Functional Description

16.4.1 General

The XBAR module has only one mode of operation, functional mode.

16.4.2 Functional Mode

The value of each mux output is XBAR_OUT[n] = XBAR_IN[SELn]. The SELn select values are configured in the XBAR_SEL registers. All muxes share the same inputs in the same order.

A subset of XBAR_OUT[*] outputs has dedicated control fields in a Crossbar Control (XBAR_CTRL) register. Control fields provide the ability to perform edge detection on the corresponding XBAR_OUT output. Edge detection in turn can optionally be used to trigger an interrupt or DMA request. The intention is that, by detecting specified edges on signals propagating through the Crossbar, interrupts or DMA requests can be triggered to perform data transfers to or from other system components.

Control fields include an edge status field (STS), an detected edge type field (EDGE), and interrupt and DMA enable fields (DEN and IEN). STSn is set to 1 when an edge consistent with EDGEn occurs on XBAR_OUT[n]. STSn is cleared by writing 1 to it. Writing 0 as no effect. See Interrupts and DMA Requests for details on the use of STSn for DMA and interrupt request generation.

16.5 Resets

The XBAR module can be reset by only a hard reset, which forces all registers to their reset state.

16.6 Clocks

All sequential functionality is controlled by the Bus Clock.

16.7 Interrupts and DMA Requests

For each XBAR_OUT[*] output with XBAR_CTRL register support, DMA or interrupt functionality can be enabled by setting the corresponding XBAR_CTRL register bit DENn or IENn to 1. DENn and IENn should not be set to 1 at the same time for the same output XBAR_OUT[n].

Setting DENn to 1 enables DMA functionality for XBAR_OUT[n]. When DMA functionality is enabled, the output DMA_REQ[n] reflects the value of STSn. Thus the DMA request asserts when the edge specified by EDGEn is detected on XBAR_OUT[n]. Also, a rising edge on DMA_ACK[n] sets STSn to zero and thus clears the DMA request. When DEN is 0, DMA_REQ[n] is held low and DMA_ACK[n] is ignored.

Setting IENn to 1 enables interrupt functionality for XBAR_OUT[n]. When interrupt functionality is enabled, the output INT_REQ[n] reflects the value of STSn. Thus the interrupt request asserts when the edge specified by EDGEDENn is detected on XBAR_OUT[n]. The interrupt request is cleared by writing a 1 to STSn. When IENn is 0, INT_REQ[n] is held low.

DENn and IENn should not be set to 1 at the same time for the same output XBAR_OUT[n].

Chapter 17
Inter-Peripheral Crossbar Switch B (XBARB): AOI Input

17.1 Introduction

For a description of the general features and functionality of this module, refer to the XBARA description.

17.2 Memory Map and Register Descriptions

This section provides information about the XBARB instance of the inter-peripheral crossbar switch. Refer to the XBARA register details for information about that instance of the module.

This XBAR module has only select registers.

In the XBAR select registers, the SELn fields select which of the shared inputs (XBAR_IN[*]) is muxed to each mux output (XBAR_OUT[*]). There is one SELn field per mux and therefore one per XBAR_OUT output. Crossbar output XBAR_OUT[n] presents the value of XBAR_IN[SELn]. Each select register contains two SELn fields. In the first select register, the LSBs contain the select field for mux 0, and the MSBs contain the select field for mux 1. The pattern repeats in subsequent select registers.

The actual signals connected to XBAR_IN and XBAR_OUT are application specific and are described in the Chip Configuration details.

XBARB memory map

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
E360	Crossbar B Select Register 0 (XBARB_SEL0)	16	R/W	0000h	17.2.1/320
E361	Crossbar B Select Register 1 (XBARB_SEL1)	16	R/W	0000h	17.2.2/320
E362	Crossbar B Select Register 2 (XBARB_SEL2)	16	R/W	0000h	17.2.3/321
E363	Crossbar B Select Register 3 (XBARB_SEL3)	16	R/W	0000h	17.2.4/321
E364	Crossbar B Select Register 4 (XBARB_SEL4)	16	R/W	0000h	17.2.5/322
E365	Crossbar B Select Register 5 (XBARB_SEL5)	16	R/W	0000h	17.2.6/322
E366	Crossbar B Select Register 6 (XBARB_SEL6)	16	R/W	0000h	17.2.7/323
E367	Crossbar B Select Register 7 (XBARB_SEL7)	16	R/W	0000h	17.2.8/323

17.2.1 Crossbar B Select Register 0 (XBARB_SEL0)

Address: E360h base + 0h offset = E360h

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	0			SEL1					0			SEL0				
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

XBARB_SEL0 field descriptions

Field	Description
15–13 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
12–8 SEL1	Input (XBARB_INn) to be muxed to XBARB_OUT1 (refer to Functional Description section for input/output assignment)
7–5 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
SEL0	Input (XBARB_INn) to be muxed to XBARB_OUT0 (refer to Functional Description section for input/output assignment)

17.2.2 Crossbar B Select Register 1 (XBARB_SEL1)

Address: E360h base + 1h offset = E361h

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	0			SEL3					0			SEL2				
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

XBARB_SEL1 field descriptions

Field	Description
15–13 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
12–8 SEL3	Input (XBARB_INn) to be muxed to XBARB_OUT3 (refer to Functional Description section for input/output assignment)
7–5 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
SEL2	Input (XBARB_INn) to be muxed to XBARB_OUT2 (refer to Functional Description section for input/output assignment)

17.2.3 Crossbar B Select Register 2 (XBARB_SEL2)

Address: E360h base + 2h offset = E362h

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	0			SEL5					0			SEL4				
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

XBARB_SEL2 field descriptions

Field	Description
15–13 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
12–8 SEL5	Input (XBARB_INn) to be muxed to XBARB_OUT5 (refer to Functional Description section for input/output assignment)
7–5 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
SEL4	Input (XBARB_INn) to be muxed to XBARB_OUT4 (refer to Functional Description section for input/output assignment)

17.2.4 Crossbar B Select Register 3 (XBARB_SEL3)

Address: E360h base + 3h offset = E363h

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	0			SEL7					0			SEL6				
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

XBARB_SEL3 field descriptions

Field	Description
15–13 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

Table continues on the next page...

XBARB_SEL3 field descriptions (continued)

Field	Description
12–8 SEL7	Input (XBARB_INn) to be muxed to XBARB_OUT7 (refer to Functional Description section for input/output assignment)
7–5 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
SEL6	Input (XBARB_INn) to be muxed to XBARB_OUT6 (refer to Functional Description section for input/output assignment)

17.2.5 Crossbar B Select Register 4 (XBARB_SEL4)

Address: E360h base + 4h offset = E364h

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	0			SEL9					0			SEL8				
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

XBARB_SEL4 field descriptions

Field	Description
15–13 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
12–8 SEL9	Input (XBARB_INn) to be muxed to XBARB_OUT9 (refer to Functional Description section for input/output assignment)
7–5 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
SEL8	Input (XBARB_INn) to be muxed to XBARB_OUT8 (refer to Functional Description section for input/output assignment)

17.2.6 Crossbar B Select Register 5 (XBARB_SEL5)

Address: E360h base + 5h offset = E365h

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	0			SEL11					0			SEL10				
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

XBARB_SEL5 field descriptions

Field	Description
15–13 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
12–8 SEL11	Input (XBARB_INn) to be muxed to XBARB_OUT11 (refer to Functional Description section for input/output assignment)

Table continues on the next page...

XBARB_SEL5 field descriptions (continued)

Field	Description
7–5 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
SEL10	Input (XBARB_INn) to be muxed to XBARB_OUT10 (refer to Functional Description section for input/output assignment)

17.2.7 Crossbar B Select Register 6 (XBARB_SEL6)

Address: E360h base + 6h offset = E366h

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	0			SEL13					0			SEL12				
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

XBARB_SEL6 field descriptions

Field	Description
15–13 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
12–8 SEL13	Input (XBARB_INn) to be muxed to XBARB_OUT13 (refer to Functional Description section for input/output assignment)
7–5 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
SEL12	Input (XBARB_INn) to be muxed to XBARB_OUT12 (refer to Functional Description section for input/output assignment)

17.2.8 Crossbar B Select Register 7 (XBARB_SEL7)

Address: E360h base + 7h offset = E367h

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	0			SEL15					0			SEL14				
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

XBARB_SEL7 field descriptions

Field	Description
15–13 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
12–8 SEL15	Input (XBARB_INn) to be muxed to XBARB_OUT15 (refer to Functional Description section for input/output assignment)
7–5 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

Table continues on the next page...

XBARB_SEL7 field descriptions (continued)

Field	Description
SEL14	Input (XBARB_INn) to be muxed to XBARB_OUT14 (refer to Functional Description section for input/ output assignment)

Chapter 18
On-Chip Clock Synthesis (OCCS)

18.1 Introduction

18.1.1 Overview

This module provides the 2X system clock frequency to the System Integration Module (SIM), which generates the various derivative system and peripheral clocks for the chip.

The on-chip clock synthesis module allows product design using several user selectable clock sources, including an 8 MHz / 400 kHz internal relaxation oscillator, a 32 kHz internal RC oscillator, an external clock input, a 4 MHz to 16 MHz external crystal oscillator, and a PLL to run up to a 100 MHz system bus frequency.

NOTE

The 8 MHz relaxation oscillator has a reduced-power standby mode in which it operates at 400 kHz. Nevertheless, in general, the oscillator is called simply the 8 MHz relaxation oscillator.

18.1.2 Features

Clock generation features include:

- PLL supporting any clock source input from 8 MHz to 50 MHz with programmable integer feedback divide (multiply factor) and maximum rated output of 400 MHz.
- PLL divide-by-2 that acts as a digital duty cycle corrector.
- Glitch free selection of input clock source.
- Glitch free selection of output clock to be any input clock source or PLL. Maximum output clock frequency is 200 MHz.
- Output clock postscaler with power-of-2 divide factors from 1 to 256.
- All input clock sources available as OCCS outputs.

Additional OCCS module features include:

- Ability to power down the 8 MHz internal relaxation oscillator.
- Ability to put the 8 MHz internal relaxation oscillator into a reduced power 400 kHz standby mode.
- Ability to power down external crystal oscillator.
- Ability to power down the 32 kHz internal RC oscillator.
- 8-bit postscaler that operates on either PLL output or, when the PLL is not in use, one of the oscillators or an external clock source.
- Ability to power down the internal PLL.
- Provides 2X master clock frequency and 2X High Speed Peripheral clock signals.
- Optional automatic switch to backup clock on loss of clock reference.
- PLL lock or loss-of-lock detection.
- Memory mapped registers for configuring the OCCS and internal clock sources.

Key features of the crystal oscillator module are:

- Supports 4 MHz to 16 MHz crystals and resonators.
- High gain option.
- Voltage and frequency filtering to guarantee clock frequency and stability.

18.2 Modes of Operation

Either an internal oscillator, crystal oscillator, or an external frequency source can be used to provide a reference clock (sys_clk_2x) to the SIM.

The 2X system clock source output from the OCCS has a maximum supported frequency of 200MHz and can be described by one of the following equations:

2X system frequency = (reference clock frequency) / (postscaler)

2X system frequency = (reference clock frequency X PLL divide factor) / (postscaler)

2X system frequency = (reference clock frequency X PLL divide factor) / (2 * postscaler)

where:

- postscaler = 1, 2, 4, 8, 16, 32, 64, 128 or 256
- PLL output divider PLL divide vector = integer from 1-128
- Reference clock frequency for use with PLL limited to 8-40MHz

The SIM is responsible for further dividing these frequencies by two, which will insure a 50% duty cycle in the system clocks.

The on-chip clock synthesis module has the following registers:

- PLL control register (CTRL)
- Divide-by register (DIVBY)
- OCCS Status Register (STAT)

- Test register (TESTR)
- Oscillator Control register (OSCTL1, OSCTL2)
- Clock Check Reference (CLKCHKR)
- Clock Check Target (CLKCHKT)
- Protect (PROT)

For more information on these registers please refer to Memory Map and Register Descriptions.

18.2.1 Internal Clock Sources

The two internal relaxation oscillator are optimized for accuracy and programmability while providing several different frequency and power saving configurations to accommodate different operating conditions. The internal oscillators have trim controls whose initialization values are determined in the factor to compensate for variability with temperature and voltage and fabrication process. They are very fast in reaching a stable frequency.

The 8MHz internal relaxation oscillator provides an 8 MHz clock at the center of its tuning range. It also supports a 400KHz standby state and a powerdown state. Frequency can be adjusted up or down using a primary trim adjustment. This is used to shift the frequency vs. temperature curve up and down to center it on 8MHz. The primary frequency trim is controlled by 10 bits. This oscillator also supports a 4-bit temperature trim factor which biases the slope of the frequency temperature curve. This is useful for reducing the maximum deviation from nominal frequency over temperature. During the reset sequence, this oscillator will be enabled by default. Application code can then switch to another clock source and power down this oscillator if desired.

The 32KHz internal RC oscillator provides a 32KHz clock at the center frequency of its tuning range. Frequency can be adjusted up or down using a 9 bit primary trim value. This is used to shift the frequency vs. temperature curve up or down to center it on 32KHz. During the reset sequence, this oscillator will be disabled by default.

Both oscillator’s are characterized in the factory and recommended trim values delivered in a reserved area in the parts flash memory and loaded into the OCCS trim registers by standard application software.

18.2.2 Loop Controlled Pierce Crystal Oscillator

The internal crystal oscillator circuit is designed to interface with an external crystal or resonator with a frequency in the range of 4-16MHz. The oscillator supports two primary modes of operation; Loop Controlled Pierce mode (LCP) with a frequency range from 4 MHz to 16 MHz and Full Swing Pierce mode (FSP) with a frequency range from 4 MHz to 16 MHz. It also supports an external clock bypass mode covered in the next section. When used to supply a source to the internal PLL, the crystal/resonator must be in the 8 MHz to 16 MHz range. Follow the crystal supplier's recommendations when selecting a crystal, since crystal parameters determine the component values required to provide maximum stability and reliable start-up. The load capacitance values used in the oscillator circuit design should include all stray layout capacitances. The crystal and associated components should be mounted as close as possible to the EXTAL and XTAL pins to minimize output distortion and start-up stabilization time.

18.2.3 External Clock Source - Crystal Oscillator Bypass Option

In this mode, an external clock is applied using the external EXTAL pin function and propagated to the oscillator's EXTAL input. The crystal oscillator is configured to propagate this input directly to the oscillator's clock output and the crystal oscillator is selected in OCCS as the clock source. Figure 18-1 illustrates how to connect an external clock circuit with an external clock source using EXTAL as the input. (In this mode the AC/DC parametrics (V_{ih}, V_{il} ...) are determined by the crystal oscillator module.) This method of external clocking is not recommended due to the frequency limitation on EXTAL. Refer to the OSC_LCP specifications for details.

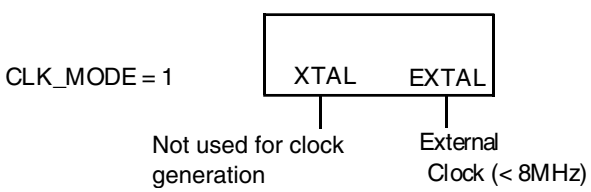


Figure 18-1. Connecting an External Clock Signal using XTAL

18.2.4 External Clock Source - CLKIN

In this mode, an external clock is applied using the external CLKIN pin function. This is propagated into the OCCS and OCCS is configured to select CLKIN as the clock source. The recommended method of connecting an external clock is given in Figure 18-2. On this device, selected GPIO can be programmed to provide the external clock input CLKIN as one of its alternate pin functions. In this mode the AC/DC parametrics (V_{ih}, V_{il} ...) are determined by the GPIO cell.

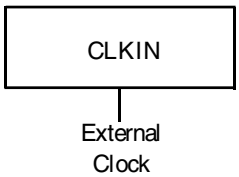


Figure 18-2. Connecting an External Clock Signal using GPIO

18.3 Block Diagram

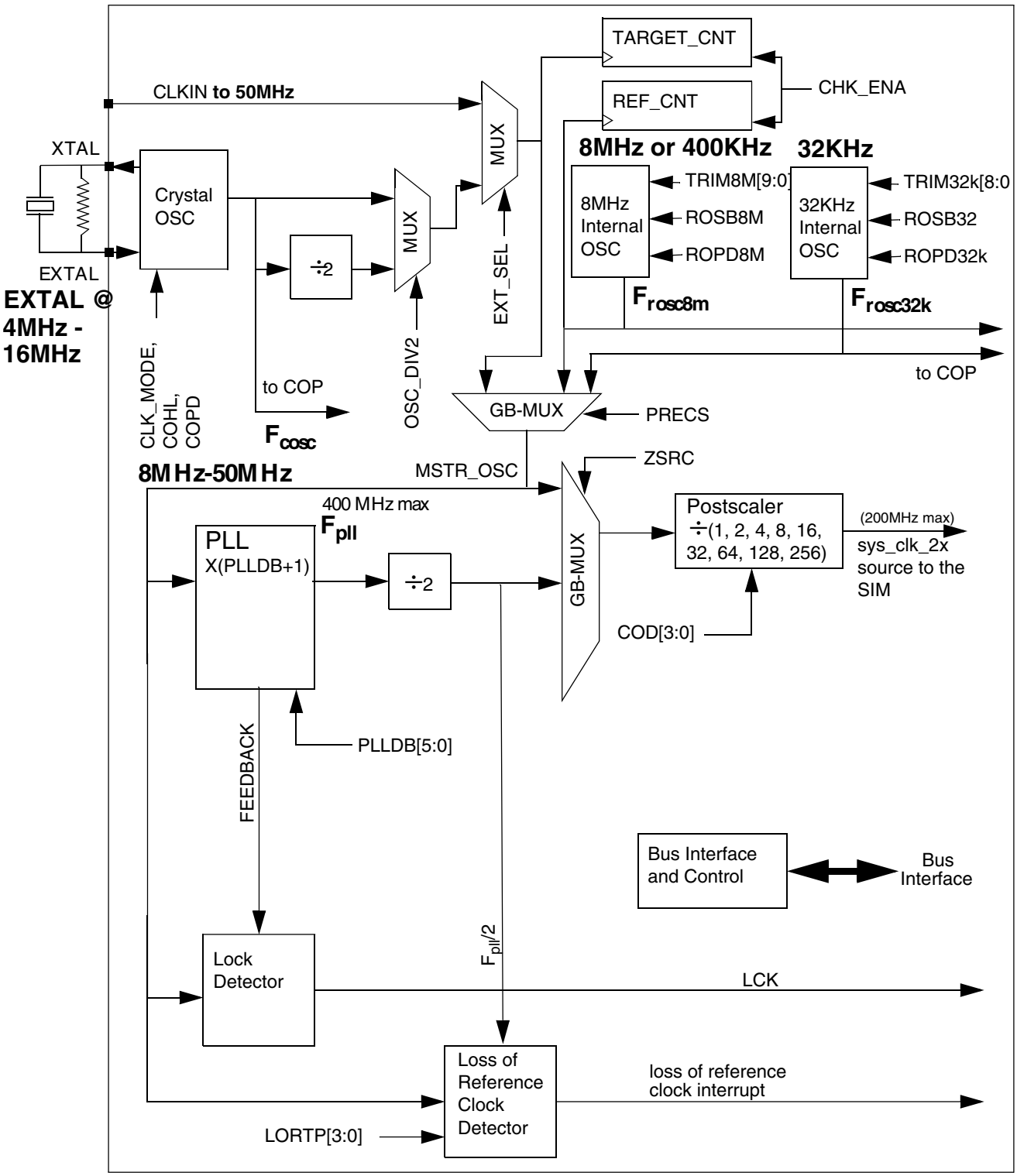


Figure 18-3. OCCS Block Diagram with Crystal Oscillator

18.4 Pin Description

18.4.1 External Clock Reference

The 8MHz relaxation oscillator is enabled in its normal operating mode at 8MHz at reset. The customer has the option of switching to an external clock reference if desired. GPIOC0 can be programmed to function as the external clock input CLKIN.

18.4.2 Oscillator IO (XTAL, EXTAL)

After reset, the user has the option of switching to the external oscillator. The oscillator inputs can be used to connect an external crystal, ceramic resonator. The EXTAL input pin function is available on the same pad, GPIOC0, as the CLKIN external clock input. Design considerations for the external clock mode of operation are discussed in Modes of Operation. The EXTAL input can optionally function as a frequency-limited external clock input. It is recommended, however, that the CLKIN pin function be used for external clocking because CLKIN does not carry these frequency limitations.

18.4.3 CLKO

This family of microcontrollers has one or more CLKO (CLKOUT) pins that can be programmed to externalize any of a number of internal clock signals. CLKO functionality exists in the System Integration Module (SIM). A number of OCCS clocks are available for use on a CLKO pin. The chip has two CLKO outputs, CLKO0 and CLKO1, so that two internal clocks can be compared to each other externally.

CAUTION

There is no defined phase relationship between the signals present on CLKO and their internal counterparts. CLKO is therefore useful for observing internal frequencies, but cannot be used to sequence data onto or off of the chip.

18.5 Memory Map and Register Descriptions

The address of a register is the sum of a base address and an address offset. The base address is defined at the system level and the address offset is defined at the module level.

OCCS memory map

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
E2B0	PLL Control Register (OCCS_CTRL)	16	R/W	0010h	18.5.1/332
E2B1	PLL Divide-By Register (OCCS_DIVBY)	16	R/W	2031h	18.5.2/334
E2B2	OCCS Status Register (OCCS_STAT)	16	R/W	0010h	18.5.3/335
E2B4	Oscillator Control Register 1 (OCCS_OSCTL1)	16	R/W	0620h	18.5.4/337
E2B5	Oscillator Control Register 2 (OCCS_OSCTL2)	16	R/W	C100h	18.5.5/338
E2B6	External Clock Check Reference (OCCS_CLKCHKR)	16	R/W	0000h	18.5.6/340
E2B7	External Clock Check Target (OCCS_CLKCHKT)	16	R	0000h	18.5.7/341
E2B8	Protection Register (OCCS_PROT)	16	R/W	0000h	18.5.8/341

18.5.1 PLL Control Register (OCCS_CTRL)

Address: E2B0h base + 0h offset = E2B0h

Bit	15	14	13	12	11	10	9	8
Read	PLLIE1		PLLIE0		LOCIE	0		
Write								
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
Read	LCKON	0		PLLPD	PRECS		0	ZSRC
Write								
Reset	0	0	0	1	0	0	0	0

OCCS_CTRL field descriptions

Field	Description
15–14 PLLIE1	PLL Interrupt Enable 1 An optional interrupt can be generated when the PLL lock status bit (LCK1) in the OCCS Status Register (STAT) changes. 00 Disable interrupt. 01 Enable interrupt on any rising edge of LCK1.

Table continues on the next page...

OCCS_CTRL field descriptions (continued)

Field	Description
	10 Enable interrupt on falling edge of LCK1. 11 Enable interrupt on any edge change of LCK1.
13–12 PLLIE0	PLL Interrupt Enable 0 An optional interrupt can be generated if the PLL lock status bit (LCK0) in the OCCS Status Register (STAT) changes. 00 Disable interrupt. 01 Enable interrupt on any rising edge of LCK0. 10 Enable interrupt on falling edge of LCK0. 11 Enable interrupt on any edge change of LCK0.
11 LOCIE	Loss of Reference Clock Interrupt Enable The loss of reference clock circuit monitors the output of the on-chip oscillator circuit. In the event of loss of reference clock, an optional interrupt can be generated. An optional interrupt can be generated if the oscillator circuit output clock is lost. 0 Interrupt disabled. 1 Interrupt enabled.
10–8 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
7 LCKON	Lock Detector On 0 Lock detector disabled 1 Lock detector enabled
6–5 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
4 PLLPD	PLL Power Down The PLL can be turned off by setting the PLLPD bit. There is a delay of four IP bus clocks between changing the bit and signaling the PLL. When the PLL is powered down, the gear shifting logic automatically switches to ZSRC = 0 to prevent a loss of the reference clock to the core. 0 PLL enabled 1 PLL powered down
3–2 PRECS	Prescaler Clock Select This bit selects between, on one hand, the external clock source or oscillator and, on the other hand, the internal relaxation oscillator. **NOTE:** Before switching to a new clock source, you must enable the new source. The relaxation oscillators are configured entirely within the OCCS. The external reference, CLKIN or external oscillator, requires configuration of the GPIO and SIM to configure the related external pads for the appropriate function as well as configuration of the OCCS itself. 00 8 MHz relaxation oscillator selected (reset value) 01 External reference selected 10 32 kHz relaxation oscillator selected 11 Reserved
1 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

Table continues on the next page...

OCCS_CTRL field descriptions (continued)

Field	Description
0 ZSRC	CLOCK Source This field determines the sys_clk_x2 source to the SIM, which generates divided-down versions of this signal for use by memories and the IP Bus. If PLLPD is set, ZSRC is automatically set to 0 to prevent a loss of the reference clock to the core. **NOTE:** Before switching to a new clock source, you must enable the new source. The PLL should be on, configured, and locked before switching to it. For extra assurance in cases where the PLL may be stressed, confirm that the PLL remains locked for a period of time before switching to it. 0 MSTR_OSC 1 PLL output divided by 2

18.5.2 PLL Divide-By Register (OCCS_DIVBY)

Address: E2B0h base + 1h offset = E2B1h

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read Write	LORTP				COD				0		PLLDB					
Reset	0	0	1	0	0	0	0	0	0	0	1	1	0	0	0	1

OCCS_DIVBY field descriptions

Field	Description
15–12 LORTP	Loss of Reference Clock Trip Point These bits control the amount of time required for the loss of reference clock interrupt to be generated. This failure detection time is `((LORTP + 1) x 10) x (reference clock period) / (PLL Multiplier / 2)`. The recommended value is LORTP ≥ 0010b. The PLL Multiplier is set by the PLLDB register.
11–8 COD	Clock Output Divide or Postscaler The PLL output clock can be divided down by a 4-bit postscaler. The input of the postscaler is a selectable clock source for the DSP core as determined by the ZSRC[1:0] bits in the control register. The output of the postscaler is guaranteed to be glitch free, even when the COD field has been changed. **NOTE:** There are special restrictions on the use of 2X high-speed peripheral clocks. 0000 Divide clock output by 1. 0001 Divide clock output by 2. 0010 Divide clock output by 4. 0011 Divide clock output by 8. 0100 Divide clock output by 16. 0101 Divide clock output by 32. 0110 Divide clock output by 64. 0111 Divide clock output by 128. 1xxx Divide clock output by 256.

Table continues on the next page...

OCCS_DIVBY field descriptions (continued)

Field	Description
7–6 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
PLLDB	PLL Divide By The output frequency of the PLL is controlled, in part, by the PLLDB[5:0] field. The value written to this field, plus one, is used by the PLL to directly multiply the input frequency and present it at its output. For example, if the input frequency is 8 MHz and the PLLDB[5:0] field is set to 49 (the default), then the PLL output frequency is 400 MHz. PLLDB settings should be limited such that the output frequency of the PLL does not exceed 400 MHz. The frequency of the primary output clock to the SIM (sys_clk_2x) depends on the setting of the ZSRC field, which selects the primary output clock source, and the COD field, which configures the postscaler. Before the divide-by value is changed, the core clock must first be switched to the MSTR_OSC clock. **NOTE:** Upon writing to the PLL Divide By register, the loss of reference detector circuit is reset.

18.5.3 OCCS Status Register (OCCS_STAT)

A PLL interrupt is generated if any of the LOLI or LOCI bits are set and the corresponding interrupt enable is set in the control register.

Address: E2B0h base + 2h offset = E2B2h

Bit	15	14	13	12	11	10	9	8
Read	LOLI1	LOLI0	LOCI	0				MON_ FAILURE
Write	w1c	w1c	w1c					
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
Read	OSC_OK	LCK1	LCK0	PLLPDN	0		ZSRCS	
Write								
Reset	0	0	0	1	0	0	0	0

OCCS_STAT field descriptions

Field	Description
15 LOLI1	PLL Lock or Loss of Lock Interrupt 1 LOLI1 shows the status of the lock detector state from the LCK1 circuit. This bit is cleared by writing a one to it. This bit will not be set (by the hardware) if the corresponding CTRL[PLLIE1] bit is cleared (set to zero).

Table continues on the next page...

OCCS_STAT field descriptions (continued)

Field	Description
	0 No lock or loss of lock event has occurred. 1 PLL lock status based on PLLIE1.
14 LOLI0	PLL Lock or Loss of Lock Interrupt 0 LOLI0 shows the status of the lock detector state from LCK0 circuit. This bit is cleared by writing a one to it. This bit will not be set (by the hardware) if the corresponding CTRL[PLLIE0] bit is cleared (set to zero). 0 No lock or loss of lock event has occurred. 1 PLL lock status based on PLLIE0.
13 LOCI	Loss of Reference Clock Interrupt LOCI shows the status of the reference clock detection circuit. This bit is cleared by writing a one to it. 0 Oscillator clock normal. 1 Loss of oscillator clock detected.
12–9 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
8 MON_FAILURE	XOSC Clock Monitor Failure Indicator 0 No clock failure, or clock monitor is disabled. 1 Clock failure detected on XOSC reference clock when clock monitor is enabled.
7 OSC_OK	OSC_OK Indicator from XOSC 0 Oscillator clock is still not stable, or XOSC is disabled. 1 XOSC OK indicator after crystal oscillator startup.
6 LCK1	PLL Lock 1 Status 0 PLL is unlocked. 1 PLL is locked (fine).
5 LCK0	PLL Lock 0 Status 0 PLL is unlocked. 1 PLL is locked (coarse).
4 PLLPDN	PLL Power Down PLL power down status is delayed by four IPbus clocks from the PLLPD bit in the control register. 0 PLL not powered down. 1 PLL powered down.
3–2 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
ZSRCS	CLOCK Source Status ZSRCS indicates the current sys_clk_x2 clock source. Because the synchronizing circuit switches the system clock source, ZSRCS takes more than one IP bus clock to indicate the new selection. 00 MSTR_OSC 01 PLL output divided by 2 1x Synchronization in progress

18.5.4 Oscillator Control Register 1 (OCCS_OSCTL1)

This register controls aspects of both the internal relaxation oscillator and the crystal/ resonator oscillator.

Address: E2B0h base + 4h offset = E2B4h

Bit	15	14	13	12	11	10	9	8
Read Write	ROPD	ROSB	COHL	CLK_MODE	OSC_DIV2	EXT_SEL	FREQ_TRIM8M	
Reset	0	0	0	0	0	1	1	0

Bit	7	6	5	4	3	2	1	0
Read Write	FREQ_TRIM8M							
Reset	0	0	1	0	0	0	0	0

OCCS_OSCTL1 field descriptions

Field	Description
15 ROPD	8 MHz Relaxation Oscillator Power Down This bit powers down the 8 MHz relaxation oscillator. To prevent a loss of clock to the core or the PLL, this bit should never be asserted while this clock source is selected by the PRECS field in the control register. 0 Relaxation oscillator enabled. 1 Relaxation oscillator powered down.
14 ROSB	8 MHz Relaxation Oscillator Standby This bit controls the power usage and gross frequency of the 8 MHz relaxation oscillator. It is reset to the more accurate but higher power state. 0 Normal mode. The relaxation oscillator output frequency is 8 MHz. 1 Standby mode. The relaxation oscillator output frequency is reduced to 400 kHz (±50%). The PLL should be disabled in this mode and MSTR_OSC should be selected as the output clock.
13 COHL	Crystal Oscillator High/Low Power Level This bit controls the power usage of the crystal oscillator. It is reset to the high power state. The low power mode should be selected to operate the crystal oscillator in Loop Controlled Pierce (LCP) mode. The high power mode should be selected to operate the crystal oscillator in Full Swing Pierce (FSP) mode or in the oscillator's external clock bypass mode. 0 High power mode. 1 Low power mode.
12 CLK_MODE	Crystal Oscillator Clock Mode This bit controls the operating mode of the crystal oscillator. When CLK_MODE is set to 1, COHL should be set to 0. External Clock Bypass Mode is not supported.

Table continues on the next page...

OCCS_OSCTL1 field descriptions (continued)

Field	Description
	NOTE: If the crystal oscillator is turned off and then turned on again, the clock should not be switched back to the oscillator until after the crystal has had time to stabilize. See the crystal data sheet to determine this time duration. 0 Crystal oscillator enabled. 1 External clock bypass mode. This enables the crystal oscillator's external clock bypass mode and allows an external clock source on the EXTAL input of the oscillator to propagate directly to the oscillator's clock output.
11 OSC_DIV2	Oscillator Divide By 2 This bit selects to the frequency range of the crystal oscillator The PRECS field should not be selecting the external clock source while changing the value of OSC_DIV2 to avoid glitches on the system clock. 0 External oscillator output is not divided. 1 External oscillator output is divided by 2 before use as MSTR_OSC.
10 EXT_SEL	External Clock In Select This bit selects the source of the external clock input. PRECS should be 0 before changing the value of EXT_SEL to avoid glitches on the system clock. 0 Use the output of the crystal oscillator as the external clock input. 1 Use CLKIN as the external clock input.
FREQ_TRIM8M	8 MHz Relaxation Oscillator Frequency Trim These bits correct part-specific variation in oscillator frequency. This control adjusts the part's average frequency over temperature up and down for the purpose of aligning it to 8 MHz. By testing the frequency of the internal clock and adjusting this factor accordingly, the accuracy of the internal clock can be improved. The reset value for this trim is 0x220, which is a center value for both gross and fine trim subfields. Refer to the crystal oscillator (IRC) specification for trim details.

18.5.5 Oscillator Control Register 2 (OCCS_OSCTL2)

This register controls aspects of both the internal relaxation oscillator and the crystal/ resonator oscillator.

Address: E2B0h base + 5h offset = E2B5h

Bit	15	14	13	12	11	10	9	8
Read Write	ROPD32K	COPD	TEMP_TRIM8M				MON_ ENABLE	FREQ_ TRIM32K
Reset	1	1	0	0	0	0	0	1

Bit	7	6	5	4	3	2	1	0
Read Write	FREQ_TRIM32K							
Reset	0	0	0	0	0	0	0	0

OCCS_OSCTL2 field descriptions

Field	Description
15 ROPD32K	32 kHz RC Oscillator Power Down This bit powers down the 32 kHz internal RC oscillator. To prevent a loss of clock to the core or the PLL, this bit should never be asserted while this clock source is selected by the PRECS field in the control register.
14 COPD	Crystal Oscillator Power Down This bit powers down the external crystal oscillator. To prevent a loss of clock to the core or the PLL, this bit should never be asserted while this clock source is selected by the PRECS field in the control register. 0 Crystal oscillator enabled. 1 Crystal oscillator powered down.
13–10 TEMP_TRIM8M	8 MHz Internal Relaxation Oscillator Temperature Trim This trim value adjusts the average slope of the oscillator's frequency as a function of temperature. The ideal shape of this function is roughly symmetrical to the X axis to minimize the WCS deviation from nominal frequency over temperature. This trim adjusts the average slope up or down by increments to accomplish this goal.
9 MON_ENABLE	XOSC Clock Monitor Enable Control This bit enables the clock monitor functionality of the XOSC.
FREQ_TRIM32K	32 kHz Internal RC Oscillator Frequency Trim These bits correct part-specific variation in oscillator frequency. This control adjusts the part's average frequency over temperature up and down for the purpose of aligning it to 32 kHz. By testing the frequency of the internal clock and incrementing or decrementing this factor accordingly, the accuracy of the internal clock can be improved. A reset sets thees bits to $100, centering the range of possible adjustment.

18.5.6 External Clock Check Reference (OCCS_CLKCHKR)

Along with the External Clock Check Target register, this register verifies the operation and relative frequency of the external clock source (target) as compared to the 8 MHz/400 kHz relaxation oscillator (reference). Verification can be performed any time the target and reference clocks are enabled; however, the greatest benefit is the ability to verify the external clock source prior to selecting it with the PRECS field.

Address: E2B0h base + 6h offset = E2B6h

Bit	15	14	13	12	11	10	9	8
Read	CHK_ENA	0						
Write								
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
Read	0	REF_CNT						
Write								
Reset	0	0	0	0	0	0	0	0

OCCS_CLKCHKR field descriptions

Field	Description
15 CHK_ENA	Check Enable This bit starts and stops the clock checking function. Allow enough time after the CLK_ENA bit is cleared to allow for two ROSC clock periods before attempting to start another verification cycle. 0 Writing a low while the clock checking operation is in progress stops the check in its current state. Reading a low after a check has been started indicates that the check operation is complete and the final values are valid in the REF_CNT and TARGET_CNT fields. 1 Writing a one clears the REF_CNT and TARGET_CNT fields and starts the clock checking function. The CLK_ENA bit remains high while the operation is in progress.
14–7 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
REF_CNT	Reference Count This count is initialized to zero on the positive transition of CHK_ENA. The test is terminated when REF_CNT equals: • 0x0080 • the number of internal 8 MHz relaxation oscillator clock cycles that have been counted **NOTE:** This counter value is not synchronized to the bus clock, and any value read while CHK_ENA is high should not be considered accurate.

18.5.7 External Clock Check Target (OCCS_CLKCHKT)

Along with the External Clock Check Reference register, this register verifies the operation and relative frequency of the external clock source (target) as compared to the 8 MHz/400 kHz relaxation oscillator (reference). Verification can be performed any time the target and reference clocks are enabled; however, the greatest benefit is the ability to verify the external clock source prior to selecting it with the PRECS field.

Address: E2B0h base + 7h offset = E2B7h

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	0					TARGET_CNT										
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

OCCS_CLKCHKT field descriptions

Field	Description
15–11 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
TARGET_CNT	CLKCHKT Target Count Number of external clock cycles that have been counted. **NOTE:** This counter value is not synchronized to the bus clock, and any value read while CHK_ENA is high should not be considered accurate. Its capacity is sufficient for verifying a 50 MHz CLKIN with sufficient room for overflow.

18.5.8 Protection Register (OCCS_PROT)

This register provides features for runaway code protection of safety-critical register fields. By choosing an appropriate subset of protection registers, you can define the trade-off between power management and protection of the OCCS operating configuration.

Flexibility is provided so that write-protection control values may themselves be optionally locked (write protected). To this end, protection controls in this register have two bit values. The right bit determines the setting of the control, and the left bit determines whether the value is locked. When a protection control is set to a locked value, it can only be altered by a chip reset which restores its default non-locked value. While a protection control remains set to non-locked values, it can be re-written to any new value.

Address: E2B0h base + 8h offset = E2B8h

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	0										FRQEP		OSCEP		PLLEP	
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

OCCS_PROT field descriptions

Field	Description
15–6 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
5–4 FRQEP	Frequency Enable Protection Enables write protection of the COD and ZSRC registers. 00 Write protection off (default). 01 Write protection on. 10 Write protection off and locked until chip reset. 11 Write protection on and locked until chip reset.
3–2 OSCEP	Oscillator Enable Protection Enables write protection of the OSCTL1, OSCTL2, and PRECS registers. 00 Write protection off (default). 01 Write protection on. 10 Write protection off and locked until chip reset. 11 Write protection on and locked until chip reset.
PLLEP	PLL Enable Protection Enables write protection of the PLLPD, LOCIE, LORTP, and PLLDB registers. When these registers are write protected (PLLPD is 0 and LOCIE is 1), the loss of reference detector cannot be disabled. 00 Write protection off (default). 01 Write protection on. 10 Write protection off and locked until chip reset. 11 Write protection on and locked until chip reset.

18.6 Functional Description

A block diagram of the OCCS module is shown in Figure 18-3.

The chip provides several alternative clock sources. This signal is referred to as MSTR_OSC in the block diagram of the OCCS. Possible clock source choices are:

- Internal 8 MHz relaxation oscillator with 400 kHz standby mode
- Internal 32 kHz oscillator
- External ceramic resonator or crystal oscillator attached to XTAL/EXTAL pad signals
- External clock source on EXTAL
- External clock source on CLKIN

The active clock source (MSTR_OSC) is selected by a glitch-free mux controlled by the setting of the PRECS field. The Internal 8 MHz relaxation oscillator is selected at reset. MSTR_OSC is used as the input to the PLL, which can be used to derive higher frequencies up to 400 MHz. When MSTR_OSC is within the spec of the PLL (8 MHz to 50 MHz), the PLL can be used to perform integer multiplication of MSTR_OSC up to a maximum 400 MHz to provide a high-speed clock source for the part. The PLL output is fed to a divide-by-2 circuit that acts as a duty cycle corrector.

The primary output clock or clock reference to the SIM module is named SYS_CLK_2X (also referred to as MSTR_2X by the SIM). The clock reference to the SIM is selected by another glitch-free mux controlled by the setting of the ZSRC field. This mux is configured to select MSTR_OSC at reset, but the clock reference can be changed using ZSRC to the PLL output divided by 2. A post-scaler is provided which can divide the clock reference by from 1 to 256 before being output to the SIM. This post-scaler can be reconfigured at any time without inducing glitches on the reference clock output to the SIM.

When ZSRC is selecting MSTR_OSC as the current clock reference, the post-scaler can be used to provide very low operating frequencies for the part. When ZSRC is selecting a PLL-based clock source, the post-scaler can be configured to provide a wide range of intermediate frequencies up to 200 MHz. The SIM uses this clock reference directly for high speed applications, divides it by two, and gates it to generate all the system and peripheral clocks that operate at a maximum of 100 MHz.

The OCCS Status Register (STAT) shows the status of the DSP core clock source. Because the synchronizing circuit changes modes to avoid any glitches, the STAT ZCLOCK source (ZSRCS) shows overlapping modes as an intermediate step. After a PLL lock is detected, the DSP core clock can be switched to the PLL by writing to the ZSRC bits in the CTRL register.

A proper sequence must be followed when reconfiguring the OCCS to ensure correct operation of the part. Before changing PRECS to change to a new clock source, the new clock source should be powered on, configured, and stable. PRECS should not be changed while ZSRC is selecting a PLL-based clock reference. This action would amount to changing the PLL input while its output is in use and would cause the clock reference to become unstable. The user should also allow for the fact that transitions of the glitch-free muxes controlled by PRECS and ZSRC require two clock periods of the old clock to disable it, and two clock periods of the new clock to enable it.

Frequencies going out of the OCCS are controlled by the post-scaler and/or the divide-by ratio within the PLL. For proper operation of the PLL, the user must keep the VCO in the PLL within its operational range of 400 MHz maximum. The output of the VCO is depicted as F_{pll} in Figure 18-3. The input frequency multiplied by the divide-by ratio is the frequency at which the VCO is running.

It is recommended when powering down, or powering up, to deselect the PLL as the clocking source. Only after lock is achieved should the PLL be selected as a valid clocking source.

Table 18-1 shows how to configure from defaults at reset to any available clock configuration.

NOTE

In the following table, the clock source refers to the PRECS selection and the clock output refers to the ZSRC selection. Configuring the clock output for direct clocking means that the PLL is not in use and ZSRC is selecting MSTR_OSC.

Table 18-1. Clock Choices Without Crystal Oscillator

Clock Source	Clock Output	Configuration Steps
8MHz Relaxation Oscillator	Direct	Default. 1. Change the COD, if desired.
32KHz Relaxation Oscillator	Direct	1. Select the 32KHz oscillator (PRECS=10). 2. Wait 6 NOPs for resynchronization. 3. The relaxation oscillator can optionally be powered down (ROPD=1). 4. Change the COD, if desired.
8MHz Relaxation Oscillator	Any PLL	1. Set PLLDB for desired multiply and enable the PLL (PLLPD=0) 2. Wait for PLL lock (LCK1=1 and LCK0=1) 3. Change ZSRC to select a PLL based source 4. Change the COD, if desired.
External Clock Source	Direct	1. The clock source (CLKIN) should be enabled in the GPIO and SIM. 2. Select CLKIN as the source clock (PRECS=01, EXT_SEL=1). 3. Wait 6 NOP's for the synchronizing circuit to change clocks. 4. The relaxation oscillator can optionally be powered down (ROPD=1). 5. Change the COD, if desired.
External Clock Source	Any PLL	1. The clock source (CLKIN) should be enabled in the GPIO and SIM. 2. Select CLKIN as the source clock (PRECS=01, EXT_SEL=1). 3. Wait 6 NOP's for the synchronizing circuit to change clocks. 4. The relaxation oscillator can optionally be powered down (ROPD=1). 5. Set PLLDB for desired multiply and enable the PLL (PLLPD=0). 6. Wait for PLL lock (LCK1=1 and LCK0=1) 7. Change ZSRC to select a PLL based output clock. 8. Change the COD, if desired.

NOTE

All of the crystal oscillator options in Table 18-2 require enabling the oscillator by setting OSCTL2[COPD] to 0.

Table 18-2. Clock Choices with Crystal Oscillator

Clock Source	Clock Output	Configuration Steps
Crystal Oscillator or Resonator in FSP mode	Direct	1. The pin functions XTAL and EXTAL should be enabled in the GPIO and SIM.

Table continues on the next page...

Table 18-2. Clock Choices with Crystal Oscillator (continued)

Clock Source	Clock Output	Configuration Steps
		2. The external clock source should be changed to the crystal oscillator (EXT_SEL=0). 3. The crystal oscillator should be powered up (CLK_MODE=0). 4. Wait for the oscillator to stabilize. 5. The crystal oscillator clock source should be selected (PRECS=01). 6. Wait 6 NOP's for the synchronizing circuit to change clocks. 7. The relaxation oscillator can optionally be powered down (ROPD=1). 8. Change the COD, if desired.
Crystal Oscillator or Resonator in FSP mode	Any PLL	1. The pin functions XTAL and EXTAL should be enabled in the GPIO and SIM. 2. The external clock source should be set to the crystal oscillator (EXT_SEL=0). 3. The crystal oscillator should be powered up (CLK_MODE=0). 4. Wait for the crystal oscillator to stabilize. 5. The crystal oscillator clock source should be selected (PRECS=01). 6. Wait 6 NOP's for the synchronizing circuit to change clocks. 7. The relaxation oscillator can optionally be powered down (ROPD=1). 8. Set PLLDB for desired multiply and enable the PLL (PLLPD=0) 9. Wait for PLL lock (LCK1=1 and LCK0=1). 10. Change ZSRC to select a PLL based output clock. 11. Change COD, if desired.
Crystal Oscillator or Resonator in LCP mode	Direct	1. The pin functions XTAL and EXTAL should be enabled in the GPIO and SIM. 2. Change the oscillator to low power mode (COHL=1). 3. The external clock source should be changed to the crystal oscillator (EXT_SEL=0). 4. The crystal oscillator should be powered up (CLK_MODE=0). 5. Wait for the oscillator to stabilize. 6. The crystal oscillator clock source should be selected (PRECS=01). 7. Wait 6 NOP's for the synchronizing circuit to change clocks. 8. The relaxation oscillator can optionally be powered down (ROPD=1). 9. Change COD, if desired.
Crystal Oscillator or Resonator in LCP mode	Any PLL	1. The pin functions XTAL and EXTAL should be enabled in the GPIO and SIM. 2. Change the oscillator to low power mode (COHL=1). 3. The external clock source should be set to the crystal oscillator (EXT_SEL=0). 4. The crystal oscillator should be powered up (CLK_MODE=0). 5. Wait for the crystal oscillator to stabilize. 6. The crystal oscillator clock source should be selected (PRECS=01). 7. Wait 6 NOP's for the synchronizing circuit to change clocks. 8. The relaxation oscillator can optionally be powered down (ROPD=1). 9. Set PLLDB for desired multiply and enable the PLL (PLLPD=0). 10. Wait for PLL lock (LCK1=1 and LCK0=1). 11. Change ZSRC to select a PLL based output clock. 12. Change COD, if desired

18.7 Relaxation Oscillators

18.7.1 Trimming Frequency on the Internal 8 MHz Relaxation Oscillator

The internal 8 MHz relaxation oscillator's frequency varies due to manufacturing process, temperature, and voltage dependencies.

The method of changing the unadjusted operating point is by changing the size of a capacitor. This capacitor value is controlled by the frequency trim factor in the OSCTL1[FREQ_TRIM8M] field. The default value for this trim factor is 220h. Each unit added or removed adjusts the output period by a fixed percentage of the unadjusted period (adding to the trim factor increases the clock period, decreasing the frequency). The trim value contains 10 bits, so the clock period of the relaxation oscillator clock can be changed to +/-40% of its unadjusted value, which is enough to cancel process variability.

To further reduce variation of frequency over temperature, a temperature trim is available in the OSCTL2[TEMP_TRIM8M] field. Its purpose is to alter the average slope of the frequency vs. temperature curve (essentially to rotate the function). This can be used to reduce the asymmetry of the curve with respect to the X axis and thus minimize the WCS deviation from nominal (8 MHz) over voltage.

Trim values for individual parts are determined at the factory, delivered in the flash memory of the part, and automatically installed by software.

A recommended way to trim the internal clock is to use the Timer module to measure the width of an input pulse on an input capture pin (this pulse must be supplied by the application and should be as long or wide as possible). Considering the prescale value of the Timer and the theoretical (zero error) frequency of the bus, the error can be calculated. This error, expressed as a percentage, can be divided by the resolution of the trim and the resulting factor added or subtracted from the frequency trim. This process should be repeated to eliminate any residual error.

18.7.2 Trimming Frequency on the Internal 32 kHz Relaxation Oscillator

Like the frequency of the 8 MHz relaxation oscillator, the frequency of the 32 kHz relaxation oscillator varies. A single 9-bit trim control is provided in the OSCTL2 register to trim the 32 kHz relaxation oscillator. Trim values for individual parts are determined at

the factory, delivered in the flash memory of the part, and automatically installed by software. Users can adjust the trim similarly to how they adjust it for the 8 MHz relaxation oscillator.

18.8 External Reference

If higher clock precision is required the chip can be operated from an external clock source.

18.9 Crystal Oscillator

The crystal oscillator is designed to operate with either an external crystal or resonator. It can also be configured in an external clock bypass mode so that it's extal input is propagated directly to its clock output. This mode is not recommended since the use of the CLKIN external pin function ovoids frequency limitations on the signal. The oscillator can be operated in either a lower power/lower frequency loop controlled pierce (LCP mode) or a higher power/ higher frequency full swing pierce mode (FSP mode) of operation.

COPD==1, CLK_MODE==x, COHL==x is powerdown mode

COPD==0, CLK_MODE==1, COHL==0 is external clock mode (ext square wave)

COPD==0, CLK_MODE==0, COHL==0 is FSP mode

COPD==0, CLK_MODE==0, COHL==1 is LCP mode

18.9.1 Switching Clock Sources

To robustly switch between the Internal Relaxation Oscillator clocks, External oscillator Clock and CLKIN, the changeover switch assumes the clocks are completely asynchronous, so a synchronizing circuit is required to make the transition. When the select input (PRECS) is changed, the switch will continue to operate off the original clock for between 1 and 2 cycles as the select input is transitioned through one side of the synchronizer. Next, the output will be held low for between 1 and 2 cycles of the new clock as the select input transitions through the other side. Then the output starts switching at the new clock's frequency. This transition guarantees that no glitches will be seen on the output even though the select input may change asynchronously to the clocks.

The unpredictability of the transition period is a necessary result of the asynchronicity. The switch automatically selects the 8MHz Internal Relaxation Oscillator clock during Reset.

Switching sources requires both clock sources to be enabled and stable. A simple flow requires:

- If switching to the crystal oscillator, make sure that XTAL and EXTAL have been enabled via GPIO and SIM and the oscillator is properly configured (COPD, COHL, CLK_MODE).
- If switching to a Relaxation Oscillator, make sure that it is powered up and configured.
- Wait for a few cycles for the clock to become Active.
- Switch clocks
- Execute 4 NOPs instructions
- Disable previous clock source (i.e. power down Relaxation Osc if Crystal is selected).

The key point to remember in this flow is that the clock source should not be switched unless the new desired clock is on and stable.

When a new DSP core clock is selected, the clock generation module will synchronize the request and select the new clock. The OCCS Status Register (STAT) shows the status of the DSP core clock source. Since the synchronizing circuit changes clock sources as to avoid any glitches, the ZSRCS bits in STAT will show overlapping modes as an intermediate step.

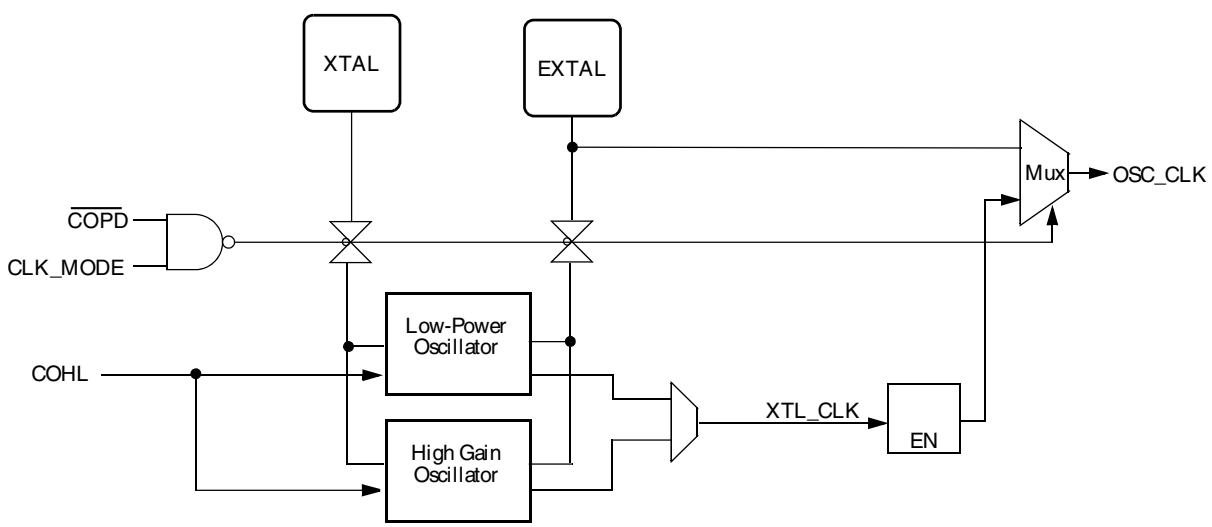


Figure 18-4. Switching Clock Sources

18.10 Phase Locked Loop

18.10.1 PLL Recommended Range of Operation

The voltage controlled oscillator (VCO) within the PLL has a characterized operating range extending from 120 MHz to 400 MHz. The output of the PLL, F_{pll}, is fed to the input of the postscaler.

For best PLL performance, the PLL source clock should be the crystal oscillator in loop controlled pierce (LCP) mode.

18.10.2 PLL Lock Time Specification

In many applications, the lock time of the PLL is the most critical PLL design parameter. Proper use of the PLL ensures the highest stability and lowest lock time.

18.10.2.1 Lock Time Definition

Typical control systems refer to the lock time as the reaction time within specified tolerances of the system to a step input. In a PLL, the step input occurs when the PLL is turned on or when it suffers a noise hit. The tolerance is usually specified as a percent of the step input or when the output settles to the desired value plus or minus a percent of the frequency change. Therefore, the reaction time is constant in this definition, regardless of the size of the step input.

When the PLL is coming from a powered down state, PLL_PDN high, to a powered up condition, it will incrementally seek its target output frequency until eventually both the gross and fine lock indicators indicate a locked condition. Refer to the device's Data Sheet for lock time specifications. Other systems refer to lock time as the time the system takes to reduce the error between the actual output and the desired output to within specified tolerances. Therefore, the lock time varies according to the original error in the output. Minor errors may be shorter or longer in many cases.

18.10.2.2 Parametric Influences on Reaction Time

Lock time is designed to be as short as possible while still providing the highest possible stability. The reaction time is not constant, however. Many factors directly and indirectly affect the lock time.

The most critical parameter affecting the reaction time of the PLL is the reference frequency, MSTR_OSC, illustrated in Figure 18-3. This frequency is the input to the phase detector and controls how often the PLL makes corrections. For stability, it is desirable for the corrections to be small and frequent. Therefore, a higher reference frequency provides optimal performance; 8MHz is minimum.

18.11 PLL Frequency Lock Detector Block

This digital block monitors the VCO output clock and sets the LCK[1:0] bits in the OCCS Status Register (STAT) based on its frequency accuracy. The lock detector is enabled with the LCKON bit of the PLL control register (CTRL), as well as the PLL_PDN bit of CTRL. Once enabled, the detector starts two counters whose outputs are periodically compared. The input clocks to these counters are the VCO output clock divided by the value in DIVBY[PLLDB], called FEEDBACK, and the PLL input clock, shown as MSTR_OSC in Figure 18-3. The period of the pulses being compared cover one whole period of each clock.

FEEDBACK and MSTR_OSC clocks are compared after 16, 32, and 64 cycles. If, after 32 cycles, the clocks match, the LCK0 bit is set to one. If, after 64 cycles of MSTR_OSC, there is the same number of MSTR_OSC clocks as FEEDBACK clocks, the LCK1 bit is also set. The LCK bit stay set until:

- Clocks fail to match
- On reset caused by LCKON, PLL_PDN
- Chip-level reset

When the circuit sets the LCK1, the two counters are reset and start the count again. The lock detector is designed so if LCK1 is reset to zero because clocks did not match, LCK0 can stay high. This provides the processor the accuracy of the two clocks with respect to each other.

18.12 Loss of Reference Clock Detector

The loss of reference clock detector is designed to generate an interrupt when the reference clock to the PLL is interrupted. An LOR interrupt should occur after a minimum time of (LORTP+1) x 10 x (reference clock period) / ((PLLDB+1) / 2). This is

the minimum time because the PLL output frequency will start to decrease as the PLL tries to track the input reference source. Figure 18-5 illustrates the general operation of the LOR detector, which relies on the fact that the phase locked loop can continue running for a time after its reference clock has been disturbed. This provides time for detection of the problem and an orderly system shutdown

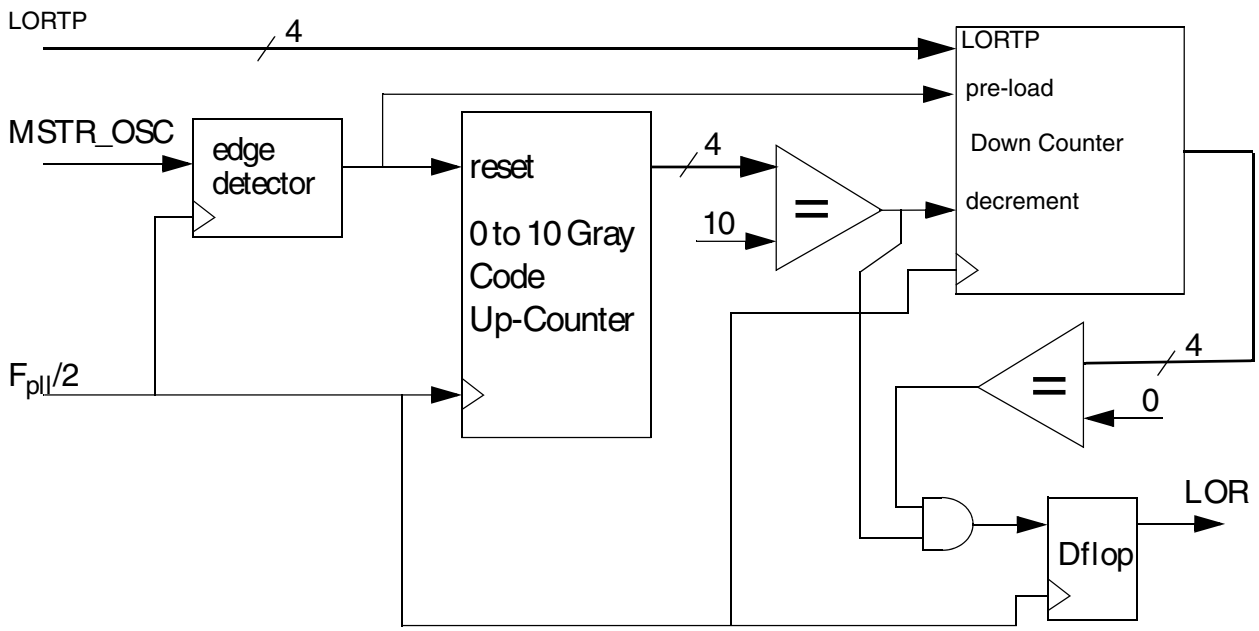


Figure 18-5. Simplified Block Diagram of the Loss Of Reference Clock Detector

18.13 Resets

18.13.1 Resets

The OCCS module is reset by a POR, an external reset, or a COP or software reset.

18.14 Clocks

Table 18-3 summarizes the various clock signals in the OCCS module. All clocks are ultimately derived from the oscillator output.

Table 18-3. Clock Summary

Clock	Source	Characteristics
IP Bus Clock	IP Bus Bridge	Derived from sys_clk and has the same frequency. Used for all peripheral register reads and writes.

Table continues on the next page...

Table 18-3. Clock Summary (continued)

Clock	Source	Characteristics
MSTR_2X	This module	Primary source for most on-chip clocks. The SIM divides this signal by two to generate the master system frequency.
FEEDBACK	PLL	FEEDBACK pin of the PLL.
F_{rosc}	Relaxation oscillator	Nominally this is 8MHz. 400KHz in standby.
F_{cosc}	crystal oscillator	Nominally this is 4-16MHz.
F_{pll}	this module	output of the PLL.

18.15 Interrupts

The interrupts listed in Table 18-4 may be OR'ed into a single processor core interrupt for some chip integrations. Inspect the interrupt table in the chip spec to determine what permutation of these interrupts are actually used.

Table 18-4. Interrupt Summary

Interrupt	Source	Description	Reference
LOLI1	STAT	Lock 1 Interrupt	CTRL register
LOLI0	STAT	Lock 2 Interrupt	CTRL register
LOCI	STAT	Loss of Reference Clock Interrupt	CTRL register

If the LOCI interrupt is enabled and the PLL is enabled then the LOCI is permitted to wakeup the system from some STOP modes.

Chapter 19
Flash Memory Controller (FMC)

19.1 Introduction

NOTE

For the chip-specific implementation details of this module's instances, see the chip configuration information.

The Flash Memory Controller (FMC) is a memory acceleration unit that provides:

- an interface between the device and the dual-bank nonvolatile memory. Bank 0 consists of program flash memory, and bank 1 consists of FlexNVM.
- buffers that can accelerate flash memory transfers.

19.1.1 Overview

The Flash Memory Controller manages the interface between the device and the dual-bank flash memory. The FMC receives status information detailing the configuration of the memory and uses this information to ensure a proper interface. The following table shows the supported read/write operations.

Flash memory type	Read	Write
Program flash memory	8-bit, 16-bit, and 32-bit reads	—A write operation to program flash memory causes a bus error.
FlexNVM used as Data flash memory	8-bit, 16-bit, and 32-bit reads	—[1]
FlexNVM and FlexRAM used as EEPROM	8-bit, 16-bit, and 32-bit reads	8-bit, 16-bit, and 32-bit writes

1. A write operation to program flash memory or to FlexNVM used as data flash memory results in a bus error.

In addition, for bank 0, the FMC provides three separate mechanisms for accelerating the interface between the device and the flash memory. A 64-bit speculation buffer can prefetch the next 64-bit flash memory location, and both a 4-way, 8-set cache and a single-entry 64-bit buffer can store previously accessed flash memory data for quick access times.

19.1.2 Features

The FMC's features include:

- Interface between the device and the dual-bank flash memory and FlexMemory:
  - 8-bit, 16-bit, and 32-bit read operations to program flash memory and FlexNVM used as data flash memory.
  - 8-bit, 16-bit, and 32-bit read and write operations to FlexNVM and FlexRAM used as EEPROM.
  - For bank 0: Read accesses to consecutive 32-bit spaces in memory return the second read data with no wait states. The memory returns 64 bits via the 32-bit bus access.
  - Crossbar master access protection for setting no access, read-only access, write-only access, or read/write access for each crossbar master.
- For bank 0: Acceleration of data transfer from program flash memory and FlexMemory to the device:
  - 64-bit prefetch speculation buffer with controls for instruction/data access per master
  - 4-way, 8-set, 64-bit line size cache for a total of thirty-two 64-bit entries with controls for replacement algorithm and lock per way
  - Single-entry buffer
  - Invalidation control for the speculation buffer and the single-entry buffer

19.2 Modes of operation

The FMC only operates when a bus master accesses the flash memory or FlexMemory.

For any device power mode where the flash memory or FlexMemory cannot be accessed, the FMC is disabled.

19.3 External signal description

The FMC has no external signals.

19.4 Memory map and register descriptions

The programming model consists of the FMC control registers and the program visible cache (data and tag/valid entries).

NOTE

Program the registers only while the flash controller is idle (for example, execute from RAM). Changing configuration settings while a flash access is in progress can lead to non-deterministic behavior.

Table 19-1. FMC register access

Registers	Read access length	Write access length
Control registers: PFAPR, PFB0CR, PFB1CR	32 bits	8, 16, or 32 bits
Cache registers	32 bits	32 bits

NOTE

Accesses to unimplemented registers within the FMC's 4 KB address space return a bus error.

The cache entries, both data and tag/valid, can be read at any time.

NOTE

System software is required to maintain memory coherence when any segment of the flash cache is programmed. For example, all buffer data associated with the reprogrammed flash should be invalidated. Accordingly, cache program visible writes must occur after a programming or erase event is completed and before the new memory image is accessed.

The cache is a 4-way, set-associative cache with 8 sets. The ways are numbered 0-3 and the sets are numbered 0-7. The following table elaborates on the tag/valid and data entries.

Table 19-2. Program visible cache registers

Cache storage	Based at offset	Contents of 32-bit read	Nomenclature	Nomenclature example
Tag	080h	13'h0, tag[18:6], 5'h0, valid	In TAGVDWxSy, x denotes the way and y denotes the set.	TAGVDW2S0 is the 13-bit tag and 1-bit valid for cache entry way 2, set 0.
Data	100h	Upper or lower longword of data	In DATAWxSyU and DATAWxSyL, x denotes the way, y denotes the set, and U and L represent upper and lower word, respectively.	DATAW1S0U represents bits [63:32] of data entry way 1, set 0, and DATAW1S0L represents bits [31:0] of data entry way 1, set 0.

NOTE

In the preceding table and in the remainder of the register information, offset and address data appears in terms of word (16-bit) addressing.

FMC memory map

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
DE00	Flash Access Protection Register (FMC_PFAPR)	32	R/W	00F0_00FFh	19.4.1/359
DE02	Flash Bank 0 Control Register (FMC_PFB0CR)	32	R/W	3002_001Fh	19.4.2/361
DE04	Flash Bank 1 Control Register (FMC_PFB1CR)	32	R/W	3000_0000h	19.4.3/363
DE80	Cache Tag Storage (FMC_TAGVDW0S0)	32	R/W	0000_0000h	19.4.4/364
DE82	Cache Tag Storage (FMC_TAGVDW0S1)	32	R/W	0000_0000h	19.4.4/364
DE84	Cache Tag Storage (FMC_TAGVDW0S2)	32	R/W	0000_0000h	19.4.4/364
DE86	Cache Tag Storage (FMC_TAGVDW0S3)	32	R/W	0000_0000h	19.4.4/364
DE88	Cache Tag Storage (FMC_TAGVDW0S4)	32	R/W	0000_0000h	19.4.4/364
DE8A	Cache Tag Storage (FMC_TAGVDW0S5)	32	R/W	0000_0000h	19.4.4/364
DE8C	Cache Tag Storage (FMC_TAGVDW0S6)	32	R/W	0000_0000h	19.4.4/364
DE8E	Cache Tag Storage (FMC_TAGVDW0S7)	32	R/W	0000_0000h	19.4.4/364
DE90	Cache Tag Storage (FMC_TAGVDW1S0)	32	R/W	0000_0000h	19.4.5/365
DE92	Cache Tag Storage (FMC_TAGVDW1S1)	32	R/W	0000_0000h	19.4.5/365
DE94	Cache Tag Storage (FMC_TAGVDW1S2)	32	R/W	0000_0000h	19.4.5/365
DE96	Cache Tag Storage (FMC_TAGVDW1S3)	32	R/W	0000_0000h	19.4.5/365
DE98	Cache Tag Storage (FMC_TAGVDW1S4)	32	R/W	0000_0000h	19.4.5/365
DE9A	Cache Tag Storage (FMC_TAGVDW1S5)	32	R/W	0000_0000h	19.4.5/365
DE9C	Cache Tag Storage (FMC_TAGVDW1S6)	32	R/W	0000_0000h	19.4.5/365
DE9E	Cache Tag Storage (FMC_TAGVDW1S7)	32	R/W	0000_0000h	19.4.5/365
DEA0	Cache Tag Storage (FMC_TAGVDW2S0)	32	R/W	0000_0000h	19.4.6/366
DEA2	Cache Tag Storage (FMC_TAGVDW2S1)	32	R/W	0000_0000h	19.4.6/366
DEA4	Cache Tag Storage (FMC_TAGVDW2S2)	32	R/W	0000_0000h	19.4.6/366

Table continues on the next page...

FMC memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
DEA6	Cache Tag Storage (FMC_TAGVDW2S3)	32	R/W	0000_0000h	19.4.6/366
DEA8	Cache Tag Storage (FMC_TAGVDW2S4)	32	R/W	0000_0000h	19.4.6/366
DEAA	Cache Tag Storage (FMC_TAGVDW2S5)	32	R/W	0000_0000h	19.4.6/366
DEAC	Cache Tag Storage (FMC_TAGVDW2S6)	32	R/W	0000_0000h	19.4.6/366
DEAE	Cache Tag Storage (FMC_TAGVDW2S7)	32	R/W	0000_0000h	19.4.6/366
DEB0	Cache Tag Storage (FMC_TAGVDW3S0)	32	R/W	0000_0000h	19.4.7/367
DEB2	Cache Tag Storage (FMC_TAGVDW3S1)	32	R/W	0000_0000h	19.4.7/367
DEB4	Cache Tag Storage (FMC_TAGVDW3S2)	32	R/W	0000_0000h	19.4.7/367
DEB6	Cache Tag Storage (FMC_TAGVDW3S3)	32	R/W	0000_0000h	19.4.7/367
DEB8	Cache Tag Storage (FMC_TAGVDW3S4)	32	R/W	0000_0000h	19.4.7/367
DEBA	Cache Tag Storage (FMC_TAGVDW3S5)	32	R/W	0000_0000h	19.4.7/367
DEBC	Cache Tag Storage (FMC_TAGVDW3S6)	32	R/W	0000_0000h	19.4.7/367
DEBE	Cache Tag Storage (FMC_TAGVDW3S7)	32	R/W	0000_0000h	19.4.7/367
DF00	Cache Data Storage (upper word) (FMC_DATAW0S0U)	32	R/W	0000_0000h	19.4.8/367
DF02	Cache Data Storage (lower word) (FMC_DATAW0S0L)	32	R/W	0000_0000h	19.4.9/368
DF04	Cache Data Storage (upper word) (FMC_DATAW0S1U)	32	R/W	0000_0000h	19.4.8/367
DF06	Cache Data Storage (lower word) (FMC_DATAW0S1L)	32	R/W	0000_0000h	19.4.9/368
DF08	Cache Data Storage (upper word) (FMC_DATAW0S2U)	32	R/W	0000_0000h	19.4.8/367
DF0A	Cache Data Storage (lower word) (FMC_DATAW0S2L)	32	R/W	0000_0000h	19.4.9/368
DF0C	Cache Data Storage (upper word) (FMC_DATAW0S3U)	32	R/W	0000_0000h	19.4.8/367
DF0E	Cache Data Storage (lower word) (FMC_DATAW0S3L)	32	R/W	0000_0000h	19.4.9/368
DF10	Cache Data Storage (upper word) (FMC_DATAW0S4U)	32	R/W	0000_0000h	19.4.8/367
DF12	Cache Data Storage (lower word) (FMC_DATAW0S4L)	32	R/W	0000_0000h	19.4.9/368
DF14	Cache Data Storage (upper word) (FMC_DATAW0S5U)	32	R/W	0000_0000h	19.4.8/367
DF16	Cache Data Storage (lower word) (FMC_DATAW0S5L)	32	R/W	0000_0000h	19.4.9/368
DF18	Cache Data Storage (upper word) (FMC_DATAW0S6U)	32	R/W	0000_0000h	19.4.8/367
DF1A	Cache Data Storage (lower word) (FMC_DATAW0S6L)	32	R/W	0000_0000h	19.4.9/368
DF1C	Cache Data Storage (upper word) (FMC_DATAW0S7U)	32	R/W	0000_0000h	19.4.8/367
DF1E	Cache Data Storage (lower word) (FMC_DATAW0S7L)	32	R/W	0000_0000h	19.4.9/368
DF20	Cache Data Storage (upper word) (FMC_DATAW1S0U)	32	R/W	0000_0000h	19.4.10/368
DF22	Cache Data Storage (lower word) (FMC_DATAW1S0L)	32	R/W	0000_0000h	19.4.11/369
DF24	Cache Data Storage (upper word) (FMC_DATAW1S1U)	32	R/W	0000_0000h	19.4.10/368
DF26	Cache Data Storage (lower word) (FMC_DATAW1S1L)	32	R/W	0000_0000h	19.4.11/369
DF28	Cache Data Storage (upper word) (FMC_DATAW1S2U)	32	R/W	0000_0000h	19.4.10/368
DF2A	Cache Data Storage (lower word) (FMC_DATAW1S2L)	32	R/W	0000_0000h	19.4.11/369
DF2C	Cache Data Storage (upper word) (FMC_DATAW1S3U)	32	R/W	0000_0000h	19.4.10/368
DF2E	Cache Data Storage (lower word) (FMC_DATAW1S3L)	32	R/W	0000_0000h	19.4.11/369
DF30	Cache Data Storage (upper word) (FMC_DATAW1S4U)	32	R/W	0000_0000h	19.4.10/368

Table continues on the next page...

FMC memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
DF32	Cache Data Storage (lower word) (FMC_DATAW1S4L)	32	R/W	0000_0000h	19.4.11/369
DF34	Cache Data Storage (upper word) (FMC_DATAW1S5U)	32	R/W	0000_0000h	19.4.10/368
DF36	Cache Data Storage (lower word) (FMC_DATAW1S5L)	32	R/W	0000_0000h	19.4.11/369
DF38	Cache Data Storage (upper word) (FMC_DATAW1S6U)	32	R/W	0000_0000h	19.4.10/368
DF3A	Cache Data Storage (lower word) (FMC_DATAW1S6L)	32	R/W	0000_0000h	19.4.11/369
DF3C	Cache Data Storage (upper word) (FMC_DATAW1S7U)	32	R/W	0000_0000h	19.4.10/368
DF3E	Cache Data Storage (lower word) (FMC_DATAW1S7L)	32	R/W	0000_0000h	19.4.11/369
DF40	Cache Data Storage (upper word) (FMC_DATAW2S0U)	32	R/W	0000_0000h	19.4.12/369
DF42	Cache Data Storage (lower word) (FMC_DATAW2S0L)	32	R/W	0000_0000h	19.4.13/370
DF44	Cache Data Storage (upper word) (FMC_DATAW2S1U)	32	R/W	0000_0000h	19.4.12/369
DF46	Cache Data Storage (lower word) (FMC_DATAW2S1L)	32	R/W	0000_0000h	19.4.13/370
DF48	Cache Data Storage (upper word) (FMC_DATAW2S2U)	32	R/W	0000_0000h	19.4.12/369
DF4A	Cache Data Storage (lower word) (FMC_DATAW2S2L)	32	R/W	0000_0000h	19.4.13/370
DF4C	Cache Data Storage (upper word) (FMC_DATAW2S3U)	32	R/W	0000_0000h	19.4.12/369
DF4E	Cache Data Storage (lower word) (FMC_DATAW2S3L)	32	R/W	0000_0000h	19.4.13/370
DF50	Cache Data Storage (upper word) (FMC_DATAW2S4U)	32	R/W	0000_0000h	19.4.12/369
DF52	Cache Data Storage (lower word) (FMC_DATAW2S4L)	32	R/W	0000_0000h	19.4.13/370
DF54	Cache Data Storage (upper word) (FMC_DATAW2S5U)	32	R/W	0000_0000h	19.4.12/369
DF56	Cache Data Storage (lower word) (FMC_DATAW2S5L)	32	R/W	0000_0000h	19.4.13/370
DF58	Cache Data Storage (upper word) (FMC_DATAW2S6U)	32	R/W	0000_0000h	19.4.12/369
DF5A	Cache Data Storage (lower word) (FMC_DATAW2S6L)	32	R/W	0000_0000h	19.4.13/370
DF5C	Cache Data Storage (upper word) (FMC_DATAW2S7U)	32	R/W	0000_0000h	19.4.12/369
DF5E	Cache Data Storage (lower word) (FMC_DATAW2S7L)	32	R/W	0000_0000h	19.4.13/370
DF60	Cache Data Storage (upper word) (FMC_DATAW3S0U)	32	R/W	0000_0000h	19.4.14/370
DF62	Cache Data Storage (lower word) (FMC_DATAW3S0L)	32	R/W	0000_0000h	19.4.15/371
DF64	Cache Data Storage (upper word) (FMC_DATAW3S1U)	32	R/W	0000_0000h	19.4.14/370
DF66	Cache Data Storage (lower word) (FMC_DATAW3S1L)	32	R/W	0000_0000h	19.4.15/371
DF68	Cache Data Storage (upper word) (FMC_DATAW3S2U)	32	R/W	0000_0000h	19.4.14/370
DF6A	Cache Data Storage (lower word) (FMC_DATAW3S2L)	32	R/W	0000_0000h	19.4.15/371
DF6C	Cache Data Storage (upper word) (FMC_DATAW3S3U)	32	R/W	0000_0000h	19.4.14/370
DF6E	Cache Data Storage (lower word) (FMC_DATAW3S3L)	32	R/W	0000_0000h	19.4.15/371
DF70	Cache Data Storage (upper word) (FMC_DATAW3S4U)	32	R/W	0000_0000h	19.4.14/370
DF72	Cache Data Storage (lower word) (FMC_DATAW3S4L)	32	R/W	0000_0000h	19.4.15/371
DF74	Cache Data Storage (upper word) (FMC_DATAW3S5U)	32	R/W	0000_0000h	19.4.14/370
DF76	Cache Data Storage (lower word) (FMC_DATAW3S5L)	32	R/W	0000_0000h	19.4.15/371
DF78	Cache Data Storage (upper word) (FMC_DATAW3S6U)	32	R/W	0000_0000h	19.4.14/370
DF7A	Cache Data Storage (lower word) (FMC_DATAW3S6L)	32	R/W	0000_0000h	19.4.15/371
DF7C	Cache Data Storage (upper word) (FMC_DATAW3S7U)	32	R/W	0000_0000h	19.4.14/370

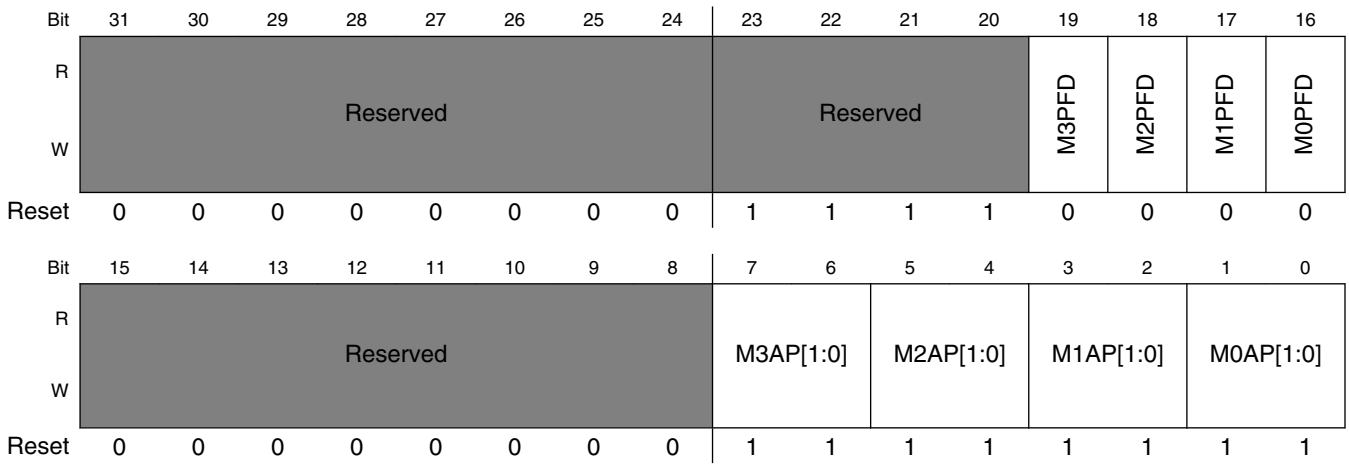
Table continues on the next page...

FMC memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
DF7E	Cache Data Storage (lower word) (FMC_DATAW3S7L)	32	R/W	0000_0000h	19.4.15/371

19.4.1 Flash Access Protection Register (FMC_PFAPR)

Address: DE00h base + 0h offset = DE00h



FMC_PFAPR field descriptions

Field	Description
31–24 Reserved	This field is reserved.
23–20 Reserved	This field is reserved. This read-only bitfield is reserved and is reset to 4'hF. Do not write to this bitfield or indeterminate results will occur.
19 M3PFD	Master 3 Prefetch Disable These bits control whether prefetching is enabled based on the logical number of the requesting crossbar switch master. This field is further qualified by the PFBnCR[BxDPE,BxIPE] bits. 0 Prefetching for this master is enabled. 1 Prefetching for this master is disabled.
18 M2PFD	Master 2 Prefetch Disable These bits control whether prefetching is enabled based on the logical number of the requesting crossbar switch master. This field is further qualified by the PFBnCR[BxDPE,BxIPE] bits. 0 Prefetching for this master is enabled. 1 Prefetching for this master is disabled.
17 M1PFD	Master 1 Prefetch Disable These bits control whether prefetching is enabled based on the logical number of the requesting crossbar switch master. This field is further qualified by the PFBnCR[BxDPE,BxIPE] bits.

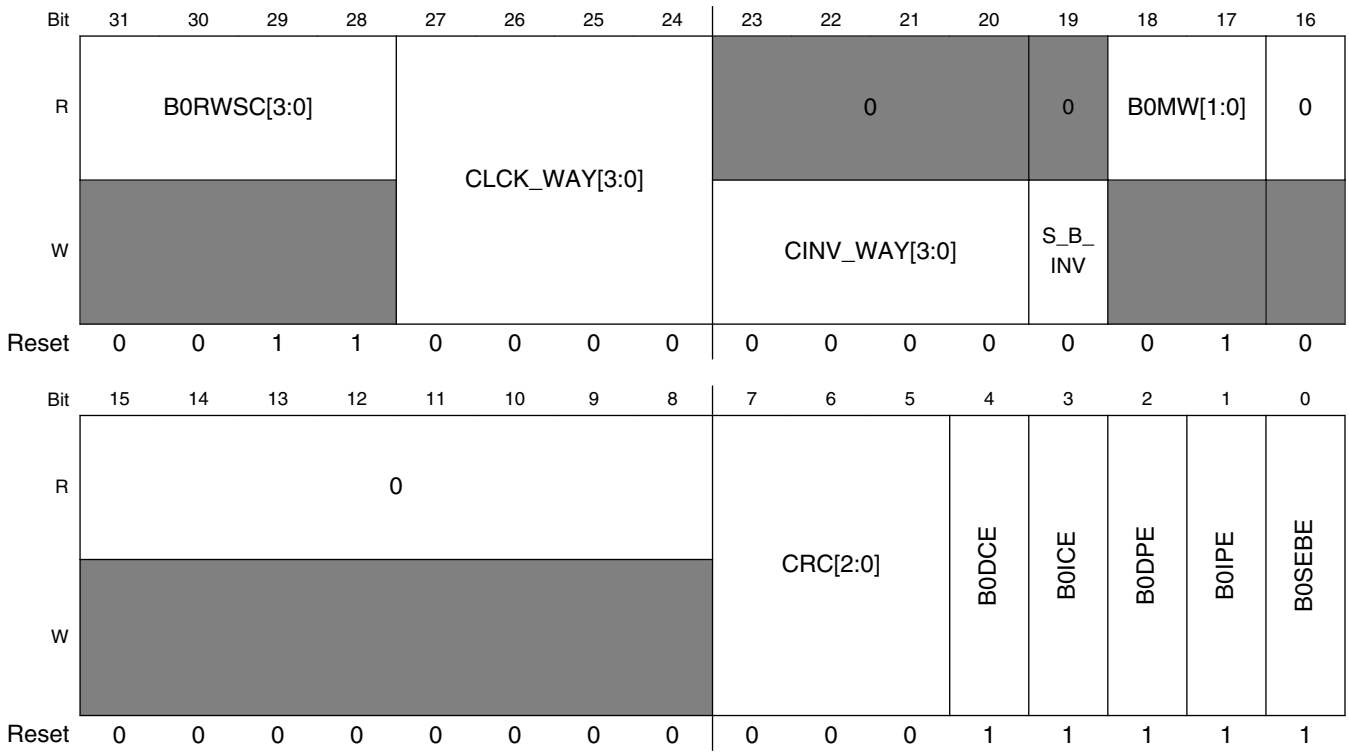
Table continues on the next page...

FMC_PFAPR field descriptions (continued)

Field	Description
	0 Prefetching for this master is enabled. 1 Prefetching for this master is disabled.
16 M0PFD	Master 0 Prefetch Disable These bits control whether prefetching is enabled based on the logical number of the requesting crossbar switch master. This field is further qualified by the PFBnCR[BxDPE,BxIPE] bits. 0 Prefetching for this master is enabled. 1 Prefetching for this master is disabled.
15–8 Reserved	This field is reserved. This read-only bitfield is reserved and is reset to zero. Do not write to this bitfield or indeterminate results will occur.
7–6 M3AP[1:0]	Master 3 Access Protection This field controls whether read and write access to the flash are allowed based on the logical master number of the requesting crossbar switch master. 00 No access may be performed by this master 01 Only read accesses may be performed by this master 10 Only write accesses may be performed by this master 11 Both read and write accesses may be performed by this master
5–4 M2AP[1:0]	Master 2 Access Protection This field controls whether read and write access to the flash are allowed based on the logical master number of the requesting crossbar switch master. 00 No access may be performed by this master 01 Only read accesses may be performed by this master 10 Only write accesses may be performed by this master 11 Both read and write accesses may be performed by this master
3–2 M1AP[1:0]	Master 1 Access Protection This field controls whether read and write access to the flash are allowed based on the logical master number of the requesting crossbar switch master. 00 No access may be performed by this master 01 Only read accesses may be performed by this master 10 Only write accesses may be performed by this master 11 Both read and write accesses may be performed by this master
M0AP[1:0]	Master 0 Access Protection This field controls whether read and write access to the flash are allowed based on the logical master number of the requesting crossbar switch master. 00 No access may be performed by this master 01 Only read accesses may be performed by this master 10 Only write accesses may be performed by this master 11 Both read and write accesses may be performed by this master

19.4.2 Flash Bank 0 Control Register (FMC_PFB0CR)

Address: DE00h base + 2h offset = DE02h



FMC_PFB0CR field descriptions

Field	Description
31–28 B0RWSC[3:0]	Bank 0 Read Wait State Control This read-only field defines the number of wait states required to access the bank 0 flash memory. The relationship between the read access time of the flash array (expressed in system clock cycles) and RWSC is defined as: Access time of flash array [system clocks] = RWSC + 1 The FMC automatically calculates this value based on the ratio of the system clock speed to the flash clock speed. For example, when this ratio is 4:1, the field's value is 3h.
27–24 CLCK_WAY[3:0]	Cache Lock Way x These bits determine if the given cache way is locked such that its contents will not be displaced by future misses. The bit setting definitions are for each bit in the field. 0 Cache way is unlocked and may be displaced 1 Cache way is locked and its contents are not displaced
23–20 CINV_WAY[3:0]	Cache Invalidate Way x

Table continues on the next page...

FMC_PFB0CR field descriptions (continued)

Field	Description
	These bits determine if the given cache way is to be invalidated (cleared). When a bit within this field is written, the corresponding cache way is immediately invalidated: the way's tag, data, and valid contents are cleared. This field always reads as zero. Cache invalidation takes precedence over locking. The cache is invalidated by system reset. System software is required to maintain memory coherency when any segment of the flash memory is programmed or erased. Accordingly, cache invalidations must occur after a programming or erase event is completed and before the new memory image is accessed. The bit setting definitions are for each bit in the field. 0 No cache way invalidation for the corresponding cache 1 Invalidate cache way for the corresponding cache: clear the tag, data, and vld bits of ways selected
19 S_B_INV	Invalidate Prefetch Speculation Buffer This bit determines if the FMC's prefetch speculation buffer and the single entry page buffer are to be invalidated (cleared). When this bit is written, the speculation buffer and single entry buffer are immediately cleared. This bit always reads as zero. 0 Speculation buffer and single entry buffer are not affected. 1 Invalidate (clear) speculation buffer and single entry buffer.
18–17 B0MW[1:0]	Bank 0 Memory Width This read-only field defines the width of the bank 0 memory. 00 32 bits 01 64 bits 10 Reserved 11 Reserved
16 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
15–8 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
7–5 CRC[2:0]	Cache Replacement Control This 3-bit field defines the replacement algorithm for accesses that are cached. 000 LRU replacement algorithm per set across all four ways 001 Reserved 010 Independent LRU with ways [0-1] for ifetches, [2-3] for data 011 Independent LRU with ways [0-2] for ifetches, [3] for data 1xx Reserved
4 B0DCE	Bank 0 Data Cache Enable This bit controls whether data references are loaded into the cache. 0 Do not cache data references. 1 Cache data references.
3 B0ICE	Bank 0 Instruction Cache Enable This bit controls whether instruction fetches are loaded into the cache.

Table continues on the next page...

FMC_PFB0CR field descriptions (continued)

Field	Description
	0 Do not cache instruction fetches. 1 Cache instruction fetches.
2 B0DPE	Bank 0 Data Prefetch Enable This bit controls whether prefetches (or speculative accesses) are initiated in response to data references. 0 Do not prefetch in response to data references. 1 Enable prefetches in response to data references.
1 B0IPE	Bank 0 Instruction Prefetch Enable This bit controls whether prefetches (or speculative accesses) are initiated in response to instruction fetches. 0 Do not prefetch in response to instruction fetches. 1 Enable prefetches in response to instruction fetches.
0 B0SEBE	Bank 0 Single Entry Buffer Enable This bit controls whether the single entry page buffer is enabled in response to flash read accesses. Its operation is independent from bank 1's cache. A high-to-low transition of this enable forces the page buffer to be invalidated. 0 Single entry buffer is disabled. 1 Single entry buffer is enabled.

19.4.3 Flash Bank 1 Control Register (FMC_PFB1CR)

This register has a format similar to that for PFB0CR, except it controls the operation of flash bank 1, and the "global" cache control fields are empty.

Address: DE00h base + 4h offset = DE04h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	B1RWSC[3:0]				0									B1MW[1:0]		0
W																
Reset	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0

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

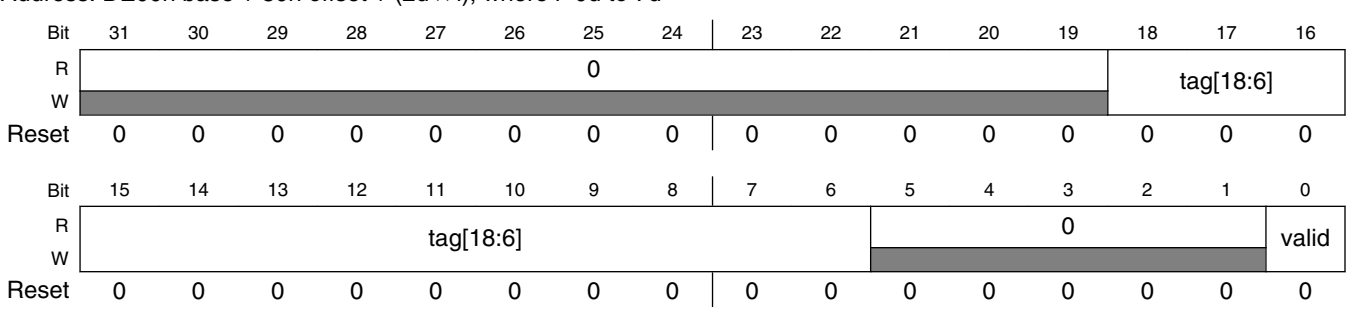
FMC_PFB1CR field descriptions

Field	Description
31–28 B1RWSC[3:0]	Bank 1 Read Wait State Control This read-only field defines the number of wait states required to access the bank 1 flash memory. The relationship between the read access time of the flash array (expressed in system clock cycles) and RWSC is defined as: Access time of flash array [system clocks] = RWSC + 1 The FMC automatically calculates this value based on the ratio of the system clock speed to the flash clock speed. For example, when this ratio is 4:1, the field's value is 3h.
27–19 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
18–17 B1MW[1:0]	Bank 1 Memory Width This read-only field defines the width of the bank 1 memory. 00 32 bits 01 64 bits 10 Reserved 11 Reserved
16 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
15–8 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
7–5 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

19.4.4 Cache Tag Storage (FMC_TAGVDW0S*n*)

The cache is a 4-way, set-associative cache with 8 sets. The ways are numbered 0-3 and the sets are numbered 0-7. In TAGVDWxSy, x denotes the way, and y denotes the set. This section represents tag/vld information for all sets in the indicated way.

Address: DE00h base + 80h offset + (2d × i), where i=0d to 7d



FMC_TAGVDW0S*n* field descriptions

Field	Description
31–19 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
18–6 tag[18:6]	13-bit tag for cache entry
5–1 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
0 valid	1-bit valid for cache entry

19.4.5 Cache Tag Storage (FMC_TAGVDW1S*n*)

The cache is a 4-way, set-associative cache with 8 sets. The ways are numbered 0-3 and the sets are numbered 0-7. In TAGVDWxSy, x denotes the way, and y denotes the set. This section represents tag/vld information for all sets in the indicated way.

Address: DE00h base + 90h offset + (2d × i), where i=0d to 7d

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0													tag[18:6]		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	tag[18:6]										0					valid
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FMC_TAGVDW1S*n* field descriptions

Field	Description
31–19 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
18–6 tag[18:6]	13-bit tag for cache entry
5–1 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
0 valid	1-bit valid for cache entry

19.4.6 Cache Tag Storage (FMC_TAGVDW2S*n*)

The cache is a 4-way, set-associative cache with 8 sets. The ways are numbered 0-3 and the sets are numbered 0-7. In TAGVDWxSy, x denotes the way, and y denotes the set. This section represents tag/vld information for all sets in the indicated way.

Address: DE00h base + A0h offset + (2d × i), where i=0d to 7d

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0													tag[18:6]		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	tag[18:6]										0					valid
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

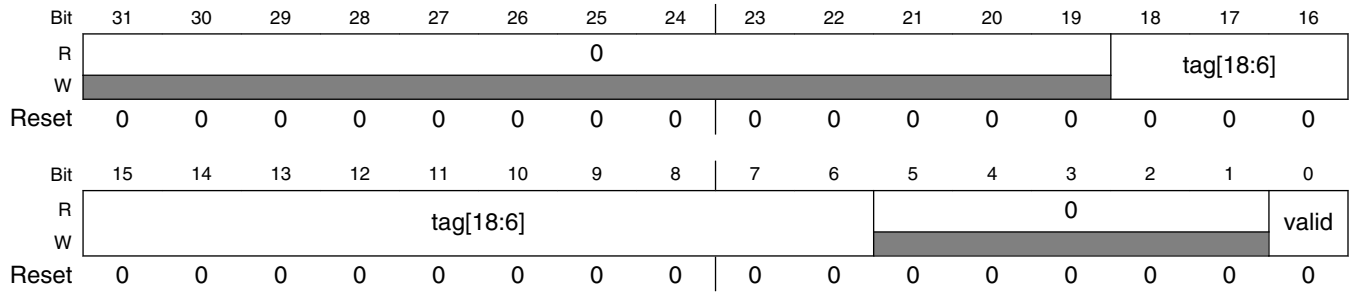
FMC_TAGVDW2S*n* field descriptions

Field	Description
31–19 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
18–6 tag[18:6]	13-bit tag for cache entry
5–1 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
0 valid	1-bit valid for cache entry

19.4.7 Cache Tag Storage (FMC_TAGVDW3S*n*)

The cache is a 4-way, set-associative cache with 8 sets. The ways are numbered 0-3 and the sets are numbered 0-7. In TAGVDWxSy, x denotes the way, and y denotes the set. This section represents tag/vld information for all sets in the indicated way.

Address: DE00h base + B0h offset + (2d × i), where i=0d to 7d



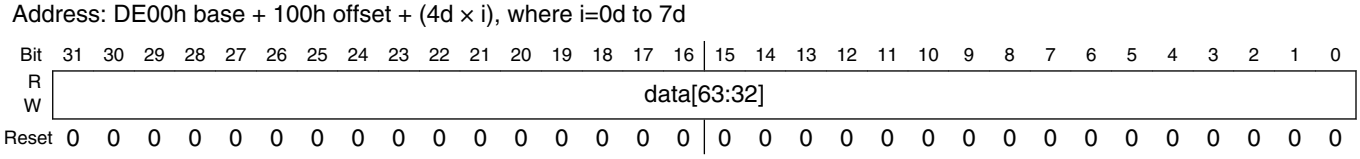
FMC_TAGVDW3S*n* field descriptions

Field	Description
31–19 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
18–6 tag[18:6]	13-bit tag for cache entry
5–1 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
0 valid	1-bit valid for cache entry

19.4.8 Cache Data Storage (upper word) (FMC_DATAW0S*n*U)

The cache of 64-bit entries is a 4-way, set-associative cache with 8 sets. The ways are numbered 0-3 and the sets are numbered 0-7. In DATAWxSyU and DATAWxSyL, x denotes the way, y denotes the set, and U and L represent upper and lower word, respectively. This section represents data for the upper word (bits [63:32]) of all sets in the indicated way.

Address: DE00h base + 100h offset + (4d × i), where i=0d to 7d



FMC_DATAW0S*n*U field descriptions

Field	Description
data[63:32]	Bits [63:32] of data entry

19.4.9 Cache Data Storage (lower word) (FMC_DATAW0S*n*L)

The cache of 64-bit entries is a 4-way, set-associative cache with 8 sets. The ways are numbered 0-3 and the sets are numbered 0-7. In DATAWxSyU and DATAWxSyL, x denotes the way, y denotes the set, and U and L represent upper and lower word, respectively. This section represents data for the lower word (bits [31:0]) of all sets in the indicated way.

Address: DE00h base + 102h offset + (4d × i), where i=0d to 7d

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	data[31:0]																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FMC_DATAW0S*n*L field descriptions

Field	Description
data[31:0]	Bits [31:0] of data entry

19.4.10 Cache Data Storage (upper word) (FMC_DATAW1S*n*U)

The cache of 64-bit entries is a 4-way, set-associative cache with 8 sets. The ways are numbered 0-3 and the sets are numbered 0-7. In DATAWxSyU and DATAWxSyL, x denotes the way, y denotes the set, and U and L represent upper and lower word, respectively. This section represents data for the upper word (bits [63:32]) of all sets in the indicated way.

Address: DE00h base + 120h offset + (4d × i), where i=0d to 7d

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	data[63:32]																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FMC_DATAW1S*n*U field descriptions

Field	Description
data[63:32]	Bits [63:32] of data entry

19.4.11 Cache Data Storage (lower word) (FMC_DATAW1S*n*L)

The cache of 64-bit entries is a 4-way, set-associative cache with 8 sets. The ways are numbered 0-3 and the sets are numbered 0-7. In DATAWxSyU and DATAWxSyL, x denotes the way, y denotes the set, and U and L represent upper and lower word, respectively. This section represents data for the lower word (bits [31:0]) of all sets in the indicated way.

Address: DE00h base + 122h offset + (4d × i), where i=0d to 7d

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	data[31:0]																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FMC_DATAW1S*n*L field descriptions

Field	Description
data[31:0]	Bits [31:0] of data entry

19.4.12 Cache Data Storage (upper word) (FMC_DATAW2S*n*U)

The cache of 64-bit entries is a 4-way, set-associative cache with 8 sets. The ways are numbered 0-3 and the sets are numbered 0-7. In DATAWxSyU and DATAWxSyL, x denotes the way, y denotes the set, and U and L represent upper and lower word, respectively. This section represents data for the upper word (bits [63:32]) of all sets in the indicated way.

Address: DE00h base + 140h offset + (4d × i), where i=0d to 7d

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	data[63:32]																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FMC_DATAW2S*n*U field descriptions

Field	Description
data[63:32]	Bits [63:32] of data entry

19.4.13 Cache Data Storage (lower word) (FMC_DATAW2S*n*L)

The cache of 64-bit entries is a 4-way, set-associative cache with 8 sets. The ways are numbered 0-3 and the sets are numbered 0-7. In DATAWxSyU and DATAWxSyL, x denotes the way, y denotes the set, and U and L represent upper and lower word, respectively. This section represents data for the lower word (bits [31:0]) of all sets in the indicated way.

Address: DE00h base + 142h offset + (4d × i), where i=0d to 7d

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	data[31:0]																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FMC_DATAW2S*n*L field descriptions

Field	Description
data[31:0]	Bits [31:0] of data entry

19.4.14 Cache Data Storage (upper word) (FMC_DATAW3S*n*U)

The cache of 64-bit entries is a 4-way, set-associative cache with 8 sets. The ways are numbered 0-3 and the sets are numbered 0-7. In DATAWxSyU and DATAWxSyL, x denotes the way, y denotes the set, and U and L represent upper and lower word, respectively. This section represents data for the upper word (bits [63:32]) of all sets in the indicated way.

Address: DE00h base + 160h offset + (4d × i), where i=0d to 7d

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	data[63:32]																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FMC_DATAW3S*n*U field descriptions

Field	Description
data[63:32]	Bits [63:32] of data entry

19.4.15 Cache Data Storage (lower word) (FMC_DATAW3S*n*L)

The cache of 64-bit entries is a 4-way, set-associative cache with 8 sets. The ways are numbered 0-3 and the sets are numbered 0-7. In DATAWxSyU and DATAWxSyL, x denotes the way, y denotes the set, and U and L represent upper and lower word, respectively. This section represents data for the lower word (bits [31:0]) of all sets in the indicated way.

Address: DE00h base + 162h offset + (4d × i), where i=0d to 7d

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	data[31:0]																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FMC_DATAW3S*n*L field descriptions

Field	Description
data[31:0]	Bits [31:0] of data entry

19.5 Functional description

The FMC is a flash acceleration unit with flexible buffers for user configuration. Besides managing the interface between the device and the flash memory and FlexMemory, the FMC can be used to restrict access from crossbar switch masters and customize the cache and buffers to provide single-cycle system-clock data-access times. Whenever a hit occurs for the prefetch speculation buffer, the cache, or the single-entry buffer, the requested data is transferred within a single system clock.

19.5.1 Default configuration

Upon system reset, the FMC is configured to provide a significant level of buffering for transfers from the flash memory:

- Crossbar masters 0, 1, 2, 3 have read access to bank 0 and bank 1.
- These masters have write access to a portion of bank 1 when FlexNVM is used with FlexRAM as EEPROM.
- For bank 0:
  - Prefetch support for data and instructions is enabled for crossbar masters 0, 1, 2, 3.

- The cache is configured for least recently used (LRU) replacement for all four ways.
- The cache is configured for data or instruction replacement.
- The single-entry buffer is enabled.

19.5.2 Configuration options

Though the default configuration provides a high degree of flash acceleration, advanced users may desire to customize the FMC buffer configurations to maximize throughput for their use cases. When reconfiguring the FMC for custom use cases, do not program the FMC's control registers while the flash memory or FlexMemory is being accessed. Instead, change the control registers with a routine executing from RAM in supervisor mode.

The FMC's cache and buffering controls within PFB0CR allow the tuning of resources to suit particular applications' needs. The cache and buffer are each controlled individually. The register controls enable buffering and prefetching per access type (instruction fetch or data reference). The cache also supports 3 types of LRU replacement algorithms:

- LRU per set across all 4 ways,
- LRU with ways [0-1] for instruction fetches and ways [2-3] for data fetches, and
- LRU with ways [0-2] for instruction fetches and way [3] for data fetches.

As an application example: if both instruction fetches and data references are accessing flash memory, then control is available to send instruction fetches, data references, or both to the cache or the single-entry buffer. Likewise, speculation can be enabled or disabled for either type of access. If both instruction fetches and data references are cached, then the cache's way resources may be divided in several ways between the instruction fetches and data references.

19.5.3 Wait states

Because the core, crossbar switch, and bus masters can be clocked at a higher frequency than the flash clock, flash memory accesses that do not hit in the speculation buffer or cache usually require wait states. The number of wait states depends on both of the following:

1. the ratio of the core clock to the flash clock, and
2. the phase relationship of the core clock and flash clock at the time the read is requested.

The ratio of the core clock to the flash clock is equal to the value of PFB0CR[B0RWSC] + 1 for bank 0 and to the value of PFB1CR[B1RWSC] + 1 for bank 1.

For example, in a system with a 4:1 core-to-flash clock ratio, a read that does not hit in the speculation buffer or the cache can take between 4 and 7 core clock cycles to complete.

- The best-case scenario is a period of 4 core clock cycles because a read from the flash memory takes 1 flash clock, which translates to 4 core clocks.
- The worst-case scenario is a period of 7 core clock cycles, consisting of 4 cycles for the read operation and 3 cycles of delay to align the core and flash clocks.
  - A delay to align the core and flash clocks might occur because you can request a read cycle on any core clock edge, but that edge does not necessarily align with a flash clock edge where the read can start.
  - In this case, the read operation is delayed by a number of core clocks equal to the core-to-flash clock ratio minus one: 4 - 1 = 3. That is, 3 additional core clock cycles are required to synchronize the clocks before the read operation can start.

All wait states and synchronization delays are handled automatically by the Flash Memory Controller. No direct user configuration is required or even allowed to set up the flash wait states.

19.5.4 Speculative reads

The FMC has a single buffer that reads ahead to the next word in the flash memory if there is an idle cycle. Speculative prefetching is programmable for each bank for instruction and/or data accesses using the B0DPE and B0IPE fields of PFB0CR. Because many code accesses are sequential, using the speculative prefetch buffer improves performance in most cases.

When speculative reads are enabled, the FMC immediately requests the next sequential address after a read completes. By requesting the next word immediately, speculative reads can help to reduce or even eliminate wait states when accessing sequential code and/or data.

For example, consider the following scenario:

- Assume a system with a 4:1 core-to-flash clock ratio and with speculative reads enabled.
- The core requests four sequential longwords in back-to-back requests, meaning there are no core cycle delays except for stalls waiting for flash memory data to be returned.
- None of the data is already stored in the cache or speculation buffer.

In this scenario, the sequence of events for accessing the four longwords is as follows:

1. The first longword read requires 4 to 7 core clocks. See Wait states for more information.

2. Due to the 64-bit data bus of the flash memory, the second longword read takes only 1 core clock because the data is already available inside the FMC. While the data for the second longword is being returned to the core, the FMC also starts reading the third and fourth longwords from the flash memory.
3. Accessing the third longword requires 3 core clock cycles. The flash memory read itself takes 4 clocks, but the first clock overlaps with the second longword read.
4. Reading the fourth longword, like the second longword, takes only 1 clock due to the 64-bit flash memory data bus.

19.6 Initialization and application information

The FMC does not require user initialization. Flash acceleration features are enabled by default.

The FMC has no visibility into flash memory erase and program cycles because the Flash Memory module manages them directly. As a result, if an application is executing flash memory commands, the FMC's cache might need to be disabled and/or flushed to prevent the possibility of returning stale data. Use the PFB0CR[CINV_WAY] field to invalidate the cache in this manner.

Chapter 20
Flash Memory Module (FTFL)

20.1 Introduction

NOTE

For the chip-specific implementation details of this module's instances, see the chip configuration information.

The flash memory module includes the following accessible memory regions:

- Program flash memory for vector space and code store
- FlexNVM for data store and additional code store
- FlexRAM for high-endurance data store or traditional RAM

Flash memory is ideal for single-supply applications, permitting in-the-field erase and reprogramming operations without the need for any external high voltage power sources.

The flash memory module includes a memory controller that executes commands to modify flash memory contents. An erased bit reads '1' and a programmed bit reads '0'. The programming operation is unidirectional; it can only move bits from the '1' state (erased) to the '0' state (programmed). Only the erase operation restores bits from '0' to '1'; bits cannot be programmed from a '0' to a '1'.

CAUTION

A flash memory location must be in the erased state before being programmed. Cumulative programming of bits (back-to-back program operations without an intervening erase) within a flash memory location is not allowed. Re-programming of existing 0s to 0 is not allowed as this overstresses the device.

The standard shipping condition for flash memory is erased with security disabled. Data loss over time may occur due to degradation of the erased ('1') states and/or programmed ('0')

states. Therefore, it is recommended that each flash block or sector be re-erased immediately prior to factory programming to ensure that the full data retention capability is achieved.

20.1.1 Features

The flash memory module includes the following features.

NOTE

See the device's Chip Configuration details for the exact amount of flash memory available on your device.

20.1.1.1 Program Flash Memory Features

- Sector size of 2 KB
- Program flash protection scheme prevents accidental program or erase of stored data
- Automated, built-in, program and erase algorithms with verify
- Section programming for faster bulk programming times
- Read access to program flash memory possible while programming or erasing data in the data flash memory or FlexRAM

20.1.1.2 FlexNVM Memory Features

When FlexNVM is partitioned for data flash memory:

- Sector size of 1 KB
- Protection scheme prevents accidental program or erase of stored data
- Automated, built-in program and erase algorithms with verify
- Section programming for faster bulk programming times
- Read access to data flash memory possible while programming or erasing data in the program flash memory

20.1.1.3 FlexRAM Features

- Memory that can be used as traditional RAM or as high-endurance EEPROM storage
- Up to 2 KB of FlexRAM configured for EEPROM or traditional RAM operations
- When configured for EEPROM:
  - Protection scheme prevents accidental program or erase of data written for EEPROM
  - Built-in hardware emulation scheme to automate EEPROM record maintenance functions
  - Programmable EEPROM data set size and FlexNVM partition code facilitating EEPROM memory endurance trade-offs
  - Supports FlexRAM aligned writes of 1, 2, or 4 bytes at a time
  - Read access to FlexRAM possible while programming or erasing data in the program or data flash memory
- When configured for traditional RAM:
  - Read and write access possible to the FlexRAM while programming or erasing data in the program or data flash memory

20.1.1.4 Other Flash Memory Module Features

- Internal high-voltage supply generator for flash memory program and erase operations
- Optional interrupt generation upon flash command completion
- Supports MCU security mechanisms which prevent unauthorized access to the flash memory contents

20.1.2 Block Diagram

The block diagram of the flash memory module is shown in the following figure.

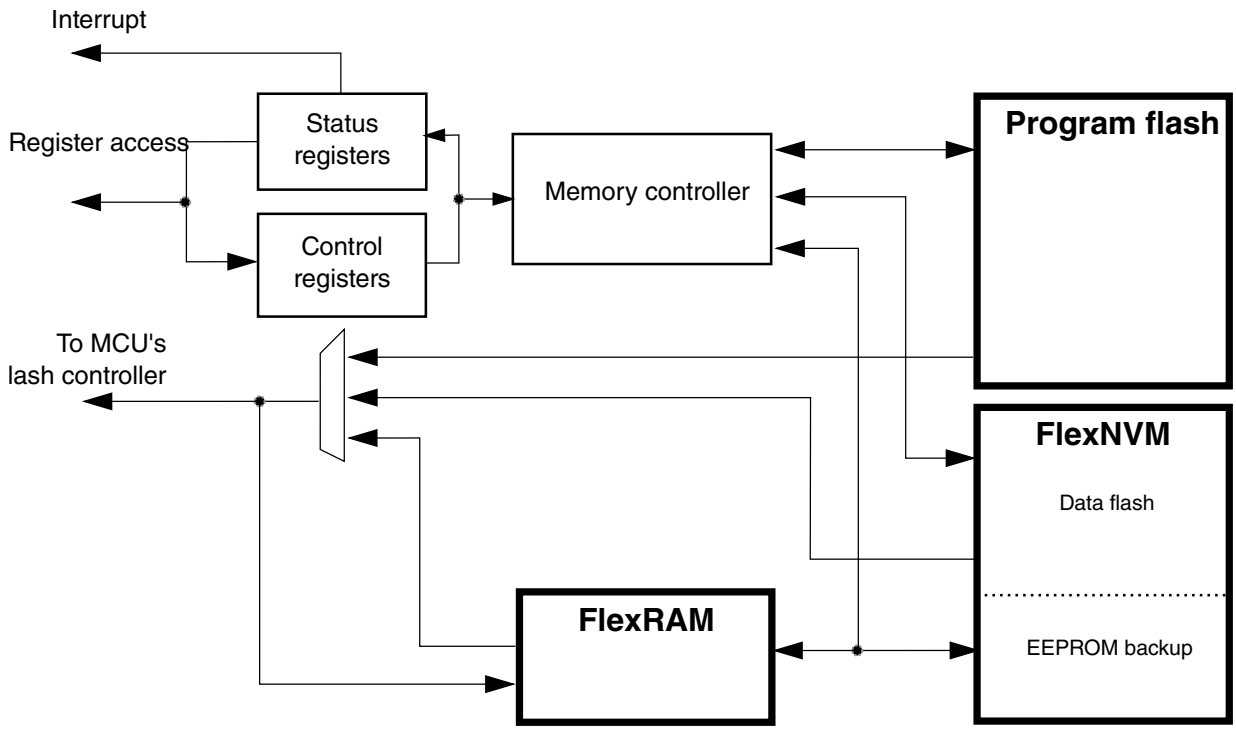


Figure 20-1. Flash Block Diagram

20.1.3 Glossary

Command write sequence — A series of MCU writes to the flash FCCOB register group that initiates and controls the execution of flash algorithms that are built into the flash memory module.

Data flash memory — Partitioned from the FlexNVM block, the data flash memory provides nonvolatile storage for user data, boot code, and additional code store.

Data flash sector — The data flash sector is the smallest portion of the data flash memory that can be erased.

EEPROM — Using a built-in filing system, the flash memory module emulates the characteristics of an EEPROM by effectively providing a high-endurance, byte-writeable (program and erase) NVM.

EEPROM backup data header — The EEPROM backup data header is comprised of a 32-bit field found in EEPROM backup data memory which contains information used by the EEPROM filing system to determine the status of a specific EEPROM backup flash sector.

EEPROM backup data record — The EEPROM backup data record is comprised of a 2-bit status field, a 14-bit address field, and a 16-bit data field found in EEPROM backup data memory which is used by the EEPROM filing system. If the status field indicates a record is valid, the data field is mirrored in the FlexRAM at a location determined by the address field.

EEPROM backup data memory — Partitioned from the FlexNVM block, EEPROM backup data memory provides nonvolatile storage for the EEPROM filing system representing data written to the FlexRAM requiring highest endurance.

EEPROM backup data sector — The EEPROM backup data sector contains one EEPROM backup data header and up to 255 EEPROM backup data records, which are used by the EEPROM filing system.

Endurance — The number of times that a flash memory location can be erased and reprogrammed.

FCCOB (Flash Common Command Object) — A group of flash registers that are used to pass command, address, data, and any associated parameters to the memory controller in the flash memory module.

Flash block — A macro within the flash memory module which provides the nonvolatile memory storage.

FlexMemory — Flash configuration that supports data flash, EEPROM, and FlexRAM.

FlexNVM Block — The FlexNVM block can be configured to be used as data flash memory, EEPROM backup flash memory, or a combination of both.

FlexRAM — The FlexRAM refers to a RAM, dedicated to the flash memory module, that can be configured to store EEPROM data or as traditional RAM. When configured for EEPROM, valid writes to the FlexRAM generate new EEPROM backup data records stored in the EEPROM backup flash memory.

Flash Memory Module — All flash blocks plus a flash management unit providing high-level control and an interface to MCU buses.

IFR — Nonvolatile information register found in each flash block, separate from the main memory array.

Longword — 32 bits of data with an aligned longword having byte-address[1:0] = 00.

NVM — Nonvolatile memory. A memory technology that maintains stored data during power-off. The flash array is an NVM using NOR-type flash memory technology.

NVM Normal Mode — An NVM mode that provides basic user access to flash memory module resources. The CPU or other bus masters initiate flash program and erase operations (or other flash commands) using writes to the FCCOB register group in the flash memory module.

NVM Special Mode — An NVM mode enabling external, off-chip access to the memory resources in the flash memory module. A reduced flash command set is available when the MCU is secured. See the Chip Configuration details for information on when this mode is used.

Phrase — 64 bits of data with an aligned phrase having byte-address[2:0] = 000.

Program flash — The program flash memory provides nonvolatile storage for vectors and code store.

Program flash Sector — The smallest portion of the program flash memory (consecutive addresses) that can be erased.

Retention — The length of time that data can be kept in the NVM without experiencing errors upon readout. Since erased (1) states are subject to degradation just like programmed (0) states, the data retention limit may be reached from the last erase operation (not from the programming time).

RWW— Read-While-Write. The ability to simultaneously read from one memory resource while commanded operations are active in another memory resource.

Section Program Buffer — Lower half of the FlexRAM allocated for storing large amounts of data for programming via the Program Section command.

Secure — An MCU state conveyed to the flash memory module as described in the Chip Configuration details for this device. In the secure state, reading and changing NVM contents is restricted.

Word — 16 bits of data with an aligned word having byte-address[0] = 0.

20.2 External Signal Description

The flash memory module contains no signals that connect off-chip.

20.3 Memory Map and Registers

This section describes the memory map and registers for the flash memory module.

Data read from unimplemented memory space in the flash memory module is undefined. Writes to unimplemented or reserved memory space (registers) in the flash memory module are ignored.

20.3.1 Flash Configuration Field Description

The program flash memory contains a 16-byte flash configuration field that stores default protection settings (loaded on reset) and security information that allows the MCU to restrict access to the flash memory module.

Flash Configuration Field Byte Address	Size (Bytes)	Field Description
0x0_0400–0x0_0407	8	Backdoor Comparison Key. Refer to Verify Backdoor Access Key Command and Unsecuring the Chip Using Backdoor Key Access.
0x0_0408–0x0_040B	4	Program flash protection bytes. Refer to the description of the Program Flash Protection Registers (FPROT0-3).
0x0_040F	1	Data flash protection byte. Refer to the description of the Data Flash Protection Register (FDPROT).
0x0_040E	1	EEPROM protection byte. Refer to the description of the EEPROM Protection Register (FEPROT).
0x0_040D	1	Flash nonvolatile option byte. Refer to the description of the Flash Option Register (FOPT).
0x0_040C	1	Flash security byte. Refer to the description of the Flash Security Register (FSEC).

20.3.2 Program Flash IFR Map

The program flash IFR is nonvolatile information memory that can be read freely, but the user has no erase and limited program capabilities (see the Read Once, Program Once, and Read Resource commands in Read Once Command, Program Once Command and Read Resource Command).

The contents of the program flash IFR are summarized in the table found here and further described in the subsequent paragraphs.

Address Range	Size (Bytes)	Field Description
0x00 – 0xBF	192	Reserved

Table continues on the next page...

Address Range	Size (Bytes)	Field Description
0xC0 – 0xFF	64	Program Once Field

20.3.2.1 Program Once Field

The Program Once Field in the program flash IFR provides 64 bytes of user data storage separate from the program flash main array. The user can program the Program Once Field one time only as there is no program flash IFR erase mechanism available to the user. The Program Once Field can be read any number of times. This section of the program flash IFR is accessed in 4-byte records using the Read Once and Program Once commands (see Read Once Command and Program Once Command).

20.3.3 Data Flash IFR Map

The data flash IFR is a 256 byte nonvolatile information memory that can be read and erased, but the user has limited program capabilities in the data flash IFR (see the Program Partition command in Program Partition Command, the Erase All Blocks command in Erase All Blocks Command, and the Read Resource command in Read Resource Command). The contents of the data flash IFR are summarized in the following table and further described in the subsequent paragraphs.

Address Range	Size (Bytes)	Field Description
0x00 – 0xFB, 0xFE – 0xFF	254	Reserved
0xFD	1	EEPROM data set size
0xFC	1	FlexNVM partition code

20.3.3.1 EEPROM Data Set Size

The EEPROM data set size byte in the data flash IFR supplies information which determines the amount of FlexRAM used in each of the available EEPROM subsystems. To program the EEESIZE value, see the Program Partition command described in Program Partition Command.

Table 20-1. EEPROM Data Set Size

Data flash IFR: 0x00FD							
7	6	5	4	3	2	1	0
1	1	1	1	EEESIZE			
	= Unimplemented or Reserved						

Table 20-2. EEPROM Data Set Size Field Description

Field	Description
7-4 Reserved	This read-only bitfield is reserved and must always be written as one.
3-0 EEESIZE	**EEPROM Size** — Encoding of the total available FlexRAM for EEPROM use. **NOTE:** EEESIZE must be 0 bytes (1111b) when the FlexNVM partition code (FlexNVM Partition Code) is set to 'No EEPROM'. '0000' = Reserved '0001' = Reserved '0010' = Reserved '0011' = 2,048 Bytes '0100' = 1,024 Bytes '0101' = 512 Bytes '0110' = 256 Bytes '0111' = 128 Bytes '1000' = 64 Bytes '1001' = 32 Bytes '1010' = Reserved '1011' = Reserved '1100' = Reserved '1101' = Reserved '1110' = Reserved '1111' = 0 Bytes

20.3.3.2 FlexNVM Partition Code

The FlexNVM Partition Code byte in the data flash IFR supplies a code which specifies how to split the FlexNVM block between data flash memory and EEPROM backup memory supporting EEPROM functions. To program the DEPART value, see the Program Partition command described in Program Partition Command.

Table 20-3. FlexNVM Partition Code

Data Flash IFR: 0x00FC							
7	6	5	4	3	2	1	0
1	1	1	1	DEPART			
	= Unimplemented or Reserved						

Table 20-4. FlexNVM Partition Code Field Description

Field	Description
7-4 Reserved	This read-only bitfield is reserved and must always be written as one.
3-0 DEPART	**FlexNVM Partition Code** — Encoding of the data flash / EEPROM backup split within the FlexNVM memory block. FlexNVM memory not partitioned for data flash will be used to store EEPROM records. 0000 = 32 KB of data flash, No EEPROM backup (No EEPROM) 0001 = 24 KB of data flash, 8 KB of EEPROM backup 0010 = 16 KB of data flash, 16 KB of EEPROM backup 0011 = No data flash, 32 KB of EEPROM backup 0100 = Reserved 0101 = Reserved 0110 = Reserved 0111 = Reserved 1000 = No data flash, 32 KB of EEPROM backup 1001 = 8 KB of data flash, 24 KB of EEPROM backup 1010 = 16 KB of data flash, 16 KB of EEPROM backup 1011 = 32 KB of data flash, No EEPROM backup (No EEPROM) 1100 = Reserved 1101 = Reserved 1110 = Reserved 1111 = Reserved (defaults to 32 KB of data flash, No EEPROM)

20.3.4 Register Descriptions

The flash memory module contains a set of memory-mapped control and status registers.

NOTE

While a command is running (FSTAT[CCIF]=0), register writes are not accepted to any register except FCNFG and FSTAT. The no-write rule is relaxed during the start-up reset sequence, prior to the initial rise of CCIF. During this initialization period the user may write any register. All register writes are also disabled (except for registers FCNFG and

FSTAT) whenever an erase suspend request is active (FCNFG[ERSSUSP]=1).

NOTE

The base address and offsets for these registers are presented in terms of bytes.

FTFL memory map

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
1_C780	Flash Status Register (FTFL_FSTAT)	8	R/W	00h	20.3.4.1/ 386
1_C781	Flash Configuration Register (FTFL_FCNFG)	8	R/W	See section	20.3.4.2/ 387
1_C782	Flash Security Register (FTFL_FSEC)	8	R	Undefined	20.3.4.3/ 389
1_C783	Flash Option Register (FTFL_FOPT)	8	R	Undefined	20.3.4.4/ 390
1_C784	Flash Common Command Object Registers (FTFL_FCCOB3)	8	R/W	00h	20.3.4.5/ 391
1_C785	Flash Common Command Object Registers (FTFL_FCCOB2)	8	R/W	00h	20.3.4.5/ 391
1_C786	Flash Common Command Object Registers (FTFL_FCCOB1)	8	R/W	00h	20.3.4.5/ 391
1_C787	Flash Common Command Object Registers (FTFL_FCCOB0)	8	R/W	00h	20.3.4.5/ 391
1_C788	Flash Common Command Object Registers (FTFL_FCCOB7)	8	R/W	00h	20.3.4.5/ 391
1_C789	Flash Common Command Object Registers (FTFL_FCCOB6)	8	R/W	00h	20.3.4.5/ 391
1_C78A	Flash Common Command Object Registers (FTFL_FCCOB5)	8	R/W	00h	20.3.4.5/ 391
1_C78B	Flash Common Command Object Registers (FTFL_FCCOB4)	8	R/W	00h	20.3.4.5/ 391
1_C78C	Flash Common Command Object Registers (FTFL_FCCOBB)	8	R/W	00h	20.3.4.5/ 391
1_C78D	Flash Common Command Object Registers (FTFL_FCCOBA)	8	R/W	00h	20.3.4.5/ 391
1_C78E	Flash Common Command Object Registers (FTFL_FCCOB9)	8	R/W	00h	20.3.4.5/ 391
1_C78F	Flash Common Command Object Registers (FTFL_FCCOB8)	8	R/W	00h	20.3.4.5/ 391
1_C790	Program Flash Protection Registers (FTFL_FPROT3)	8	R/W	Undefined	20.3.4.6/ 392

Table continues on the next page...

FTFL memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
1_C791	Program Flash Protection Registers (FTFL_FPROT2)	8	R/W	Undefined	20.3.4.6/ 392
1_C792	Program Flash Protection Registers (FTFL_FPROT1)	8	R/W	Undefined	20.3.4.6/ 392
1_C793	Program Flash Protection Registers (FTFL_FPROT0)	8	R/W	Undefined	20.3.4.6/ 392
1_C796	EEPROM Protection Register (FTFL_FEPROT)	8	R/W	Undefined	20.3.4.7/ 394
1_C797	Data Flash Protection Register (FTFL_FDPROT)	8	R/W	Undefined	20.3.4.8/ 395

20.3.4.1 Flash Status Register (FTFL_FSTAT)

The FSTAT register reports the operational status of the flash memory module.

The CCIF, RDCOLERR, ACCERR, and FPVIOL bits are readable and writable. The MGSTAT0 bit is read only. The unassigned bits read 0 and are not writable.

NOTE

When set, the Access Error (ACCERR) and Flash Protection Violation (FPVIOL) bits in this register prevent the launch of any more commands or writes to the FlexRAM (when EEERDY is set) until the flag is cleared (by writing a one to it).

Address: 1_C780h base + 0h offset = 1_C780h

Bit	7	6	5	4	3	2	1	0
Read	CCIF	RDCOLERR	ACCERR	FPVIOL	0			MGSTAT0
Write	w1c	w1c	w1c	w1c				
Reset	0	0	0	0	0	0	0	0

FTFL_FSTAT field descriptions

Field	Description
7 CCIF	Command Complete Interrupt Flag Indicates that a flash command or EEPROM file system operation has completed. The CCIF flag is cleared by writing a 1 to CCIF to launch a command, and CCIF stays low until command completion or command violation. The CCIF flag is also cleared by a successful write to FlexRAM while enabled for EEE, and CCIF stays low until the EEPROM file system has created the associated EEPROM data record. CCIF is reset to 0 but is set to 1 by the memory controller at the end of the reset initialization sequence. Depending on how quickly the read occurs after reset release, the user may or may not see the 0 hardware reset value.

Table continues on the next page...

FTFL_FSTAT field descriptions (continued)

Field	Description
	0 Flash command or EEPROM file system operation in progress 1 Flash command or EEPROM file system operation has completed
6 RDCOLERR	Flash Read Collision Error Flag Indicates that the MCU attempted a read from a flash memory resource that was being manipulated by a flash command (CCIF=0). Any simultaneous access is detected as a collision error by the block arbitration logic. The read data in this case cannot be guaranteed. The RDCOLERR bit is cleared by writing a 1 to it. Writing a 0 to RDCOLERR has no effect. 0 No collision error detected 1 Collision error detected
5 ACCERR	Flash Access Error Flag Indicates an illegal access has occurred to a flash memory resource caused by a violation of the command write sequence or issuing an illegal flash command. While ACCERR is set, the CCIF flag cannot be cleared to launch a command. The ACCERR bit is cleared by writing a 1 to it. Writing a 0 to the ACCERR bit has no effect. 0 No access error detected 1 Access error detected
4 FPVIOL	Flash Protection Violation Flag Indicates an attempt was made to program or erase an address in a protected area of program flash or data flash memory during a command write sequence or a write was attempted to a protected area of the FlexRAM while enabled for EEPROM. While FPVIOL is set, the CCIF flag cannot be cleared to launch a command. The FPVIOL bit is cleared by writing a 1 to it. Writing a 0 to the FPVIOL bit has no effect. 0 No protection violation detected 1 Protection violation detected
3–1 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
0 MGSTAT0	Memory Controller Command Completion Status Flag The MGSTAT0 status flag is set if an error is detected during execution of a flash command or during the flash reset sequence. As a status flag, this field cannot (and need not) be cleared by the user like the other error flags in this register. The value of the MGSTAT0 bit for "command-N" is valid only at the end of the "command-N" execution when CCIF=1 and before the next command has been launched. At some point during the execution of "command-N+1," the previous result is discarded and any previous error is cleared.

20.3.4.2 Flash Configuration Register (FTFL_FCNFG)

This register provides information on the current functional state of the flash memory module.

The erase control bits (ERSAREQ and ERSSUSP) have write restrictions. PFLSH, RAMRDY, and EEERDY are read-only status bits . The unassigned bits read as noted and are not writable. The reset values for the PFLASH, RAMRDY , and EEERDY bits are determined during the reset sequence.

Address: 1_C780h base + 1h offset = 1_C781h

Bit	7	6	5	4	3	2	1	0
Read	CCIE	RDCOLLIE	ERSAREQ	ERSSUSP	0	PFLSH	RAMRDY	EEERDY
Write								
Reset	0	0	0	0	0	*	*	*

* Notes:
- PFLSH field: Device specific value
- RAMRDY field: Device specific value
- EEERDY field: Device specific value

FTFL_FCNFG field descriptions

Field	Description
7 CCIE	Command Complete Interrupt Enable Controls interrupt generation when a flash command completes. 0 Command complete interrupt disabled 1 Command complete interrupt enabled. An interrupt request is generated whenever the FSTAT[CCIF] flag is set.
6 RDCOLLIE	Read Collision Error Interrupt Enable Controls interrupt generation when a flash memory read collision error occurs. 0 Read collision error interrupt disabled 1 Read collision error interrupt enabled. An interrupt request is generated whenever a flash memory read collision error is detected (see the description of FSTAT[RDCOLERR]).
5 ERSAREQ	Erase All Request Issues a request to the memory controller to execute the Erase All Blocks command and release security. ERSAREQ is not directly writable but is under indirect user control. Refer to the device's Chip Configuration details on how to request this command. ERSAREQ sets when an erase all request is triggered external to the flash memory module and CCIF is set (no command is currently being executed). ERSAREQ is cleared by the flash memory module when the operation completes. 0 No request or request complete 1 Request to: 1. run the Erase All Blocks command, 2. verify the erased state, 3. program the security byte in the Flash Configuration Field to the unsecure state, and 4. release MCU security by setting the FSEC[SEC] field to the unsecure state.
4 ERSSUSP	Erase Suspend Allows the user to suspend (interrupt) the Erase Flash Sector command while it is executing. 0 No suspend requested 1 Suspend the current Erase Flash Sector command execution.
3 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
2 PFLSH	Flash memory configuration

Table continues on the next page...

FTFL_FCNFG field descriptions (continued)

Field	Description
	0 Flash memory module configured for FlexMemory that supports data flash and/or EEPROM. 1 Reserved.
1 RAMRDY	RAM Ready This flag indicates the current status of the FlexRAM . The state of the RAMRDY flag is normally controlled by the Set FlexRAM Function command. During the reset sequence, the RAMRDY flag is cleared if the FlexNVM block is partitioned for EEPROM and is set if the FlexNVM block is not partitioned for EEPROM. The RAMRDY flag is cleared if the Program Partition command is run to partition the FlexNVM block for EEPROM. The RAMRDY flag sets after completion of the Erase All Blocks command or execution of the erase-all operation triggered external to the flash memory module. 0 FlexRAM is not available for traditional RAM access. 1 FlexRAM is available as traditional RAM only; writes to the FlexRAM do not trigger EEPROM operations.
0 EEERDY	This flag indicates if the EEPROM backup data has been copied to the FlexRAM and is therefore available for read access. During the reset sequence, the EEERDY flag will remain cleared while CCIF is clear and will only set if the FlexNVM block is partitioned for EEPROM. 0 FlexRAM is not available for EEPROM operation. 1 FlexRAM is available for EEPROM operations where: • reads from the FlexRAM return data previously written to the FlexRAM in EEPROM mode and • writes to the FlexRAM clear EEERDY and launch an EEPROM operation to store the written data in the FlexRAM and EEPROM backup.

20.3.4.3 Flash Security Register (FTFL_FSEC)

This read-only register holds all bits associated with the security of the MCU and flash memory module.

During the reset sequence, the register is loaded with the contents of the flash security byte in the Flash Configuration Field located in program flash memory. The flash basis for the values is signified by X in the reset value.

Address: 1_C780h base + 2h offset = 1_C782h

Bit	7	6	5	4	3	2	1	0
Read	KEYEN		MEEN		FSLACC		SEC	
Write								
Reset	x*	x*	x*	x*	x*	x*	x*	x*

* Notes:
- x = Undefined at reset.

FTFL_FSEC field descriptions

Field	Description
7–6 KEYEN	Backdoor Key Security Enable Enables or disables backdoor key access to the flash memory module. 00 Backdoor key access disabled 01 Backdoor key access disabled (preferred KEYEN state to disable backdoor key access) 10 Backdoor key access enabled 11 Backdoor key access disabled
5–4 MEEN	Mass Erase Enable Bits Enables and disables mass erase capability of the flash memory module. The state of this field is relevant only when SEC is set to secure outside of NVM Normal Mode. When SEC is set to unsecure, the MEEN setting does not matter. 00 Mass erase is enabled 01 Mass erase is enabled 10 Mass erase is disabled 11 Mass erase is enabled
3–2 FSLACC	Factory Security Level Access Code Enables or disables access to the flash memory contents during returned part failure analysis at NXP. When SEC is secure and FSLACC is denied, access to the program flash contents is denied and any failure analysis performed by NXP factory test must begin with a full erase to unsecure the part. When access is granted (SEC is unsecure, or SEC is secure and FSLACC is granted), NXP factory testing has visibility of the current flash contents. The state of the FSLACC bits is only relevant when SEC is set to secure. When SEC is set to unsecure, the FSLACC setting does not matter. 00 NXP factory access granted 01 NXP factory access denied 10 NXP factory access denied 11 NXP factory access granted
SEC	Flash Security Defines the security state of the MCU. In the secure state, the MCU limits access to flash memory module resources. The limitations are defined per device and are detailed in the Chip Configuration details. If the flash memory module is unsecured using backdoor key access, SEC is forced to 10b. 00 MCU security status is secure. 01 MCU security status is secure. 10 MCU security status is unsecure. (The standard shipping condition of the flash memory module is unsecure.) 11 MCU security status is secure.

20.3.4.4 Flash Option Register (FTFL_FOPT)

The flash option register allows the MCU to customize its operations by examining the state of these read-only bits, which are loaded from NVM at reset. The function of the bits is defined in the device's Chip Configuration details.

All bits in the register are read-only .

During the reset sequence, the register is loaded from the flash nonvolatile option byte in the Flash Configuration Field located in program flash memory. The flash basis for the values is signified by X in the reset value.

Address: 1_C780h base + 3h offset = 1_C783h

Bit	7	6	5	4	3	2	1	0
Read	OPT							
Write								
Reset	x*	x*	x*	x*	x*	x*	x*	x*

* Notes:
- x = Undefined at reset.

FTFL_FOPT field descriptions

Field	Description
OPT	Nonvolatile Option These bits are loaded from flash to this register at reset. Refer to the device's Chip Configuration details for the definition and use of these bits.

20.3.4.5 Flash Common Command Object Registers (FTFL_FCCOB*n*)

The FCCOB register group provides 12 bytes for command codes and parameters. The individual bytes within the set append a 0-B hex identifier to the FCCOB register name: FCCOB0, FCCOB1, ..., FCCOBB.

Address: 1_C780h base + 4h offset + (1d × i), where i=0d to 11d

Bit	7	6	5	4	3	2	1	0
Read Write	CCOBn							
Reset	0	0	0	0	0	0	0	0

FTFL_FCCOB*n* field descriptions

Field	Description
CCOBn	The FCCOB register provides a command code and relevant parameters to the memory controller. The individual registers that compose the FCCOB data set can be written in any order, but you must provide all needed values, which vary from command to command. First, set up all required FCCOB fields and then initiate the command's execution by writing a 1 to the FSTAT[CCIF] bit. This clears the CCIF bit, which locks all FCCOB parameter fields and they cannot be changed by the user until the command completes (CCIF returns to 1). No command buffering or queueing is provided; the next command can be loaded only after the current command completes.

FTFL_FCCOB*n* field descriptions (continued)

Field	Description
	Some commands return information to the FCCOB registers. Any values returned to FCCOB are available for reading after the FSTAT[CCIF] flag returns to 1 by the memory controller. The following table shows a generic flash command format. The first FCCOB register, FCCOB0, always contains the command code. This 8-bit value defines the command to be executed. The command code is followed by the parameters required for this specific flash command, typically an address and/or data values. **NOTE:** The command parameter table is written in terms of FCCOB Number (which is equivalent to the byte number). This number is a reference to the FCCOB register name and is not the register address.

FCCOB Number	Typical Command Parameter Contents [7:0]
0	FCMD (a code that defines the flash command)
1	Flash address [23:16]
2	Flash address [15:8]
3	Flash address [7:0]
4	Data Byte 0
5	Data Byte 1
6	Data Byte 2
7	Data Byte 3
8	Data Byte 4
9	Data Byte 5
A	Data Byte 6
B	Data Byte 7

FCCOB Endianness and Multi-Byte Access :

The FCCOB register group uses a big endian addressing convention. For all command parameter fields larger than 1 byte, the most significant data resides in the lowest FCCOB register number. The FCCOB register group may be read and written as individual bytes, aligned words (2 bytes) or aligned longwords (4 bytes).

20.3.4.6 Program Flash Protection Registers (FTFL_FPROT*n*)

The FPROT registers define which program flash regions are protected from program and erase operations. Protected flash regions cannot have their content changed; that is, these regions cannot be programmed and cannot be erased by any flash command. Unprotected regions can be changed by program and erase operations.

The four FPROT registers allow 32 protectable regions. Each bit protects a 1/32 region of the program flash memory . The bitfields are defined in each register as follows:

Program flash protection register	Program flash protection bits
FPROT0	PROT[31:24]
FPROT1	PROT[23:16]
FPROT2	PROT[15:8]
FPROT3	PROT[7:0]

During the reset sequence, the FPROT registers are loaded with the contents of the program flash protection bytes in the Flash Configuration Field as indicated in the following table.

Program flash protection register	Flash Configuration Field offset address
FPROT0	0x000B
FPROT1	0x000A
FPROT2	0x0009
FPROT3	0x0008

To change the program flash protection that is loaded during the reset sequence, unprotect the sector of program flash memory that contains the Flash Configuration Field. Then, reprogram the program flash protection byte.

Address: 1_C780h base + 10h offset + (1d × i), where i=0d to 3d

Bit	7	6	5	4	3	2	1	0
Read Write	PROT							
Reset	x*	x*	x*	x*	x*	x*	x*	x*

* Notes:
- x = Undefined at reset.

FTFL_FPROT*n* field descriptions

Field	Description
PROT	Program Flash Region Protect Each program flash region can be protected from program and erase operations by setting the associated PROT bit. **In NVM Normal mode:** The protection can only be increased, meaning that currently unprotected memory can be protected, but currently protected memory cannot be unprotected. Since unprotected regions are marked with a 1 and protected regions use a 0, only writes changing 1s to 0s are accepted. This 1-to-0 transition check is performed on a bit-by-bit basis. Those FPROT bits with 1-to-0 transitions are accepted while all bits with 0-to-1 transitions are ignored. **In NVM Special mode:** All bits of FPROT are writable without restriction. Unprotected areas can be protected and protected areas can be unprotected. **Restriction:** The user must never write to any FPROT register while a command is running (CCIF=0). Trying to alter data in any protected area in the program flash memory results in a protection violation error and sets the FSTAT[FPVIOL] bit. A full block erase of a program flash block is not possible if it contains any protected region.

FTFL_FPROT*n* field descriptions (continued)

Field	Description
	Each bit in the 32-bit protection register represents 1/32 of the total program flash. 0 Program flash region is protected. 1 Program flash region is not protected

20.3.4.7 EEPROM Protection Register (FTFL_FEPROT)

The FEPROT register defines which EEPROM regions of the FlexRAM are protected against program and erase operations. Protected EEPROM regions cannot have their content changed by writing to it. Unprotected regions can be changed by writing to the FlexRAM.

Address: 1_C780h base + 16h offset = 1_C796h

Bit	7	6	5	4	3	2	1	0
Read Write	EPROT							
Reset	x*	x*	x*	x*	x*	x*	x*	x*

* Notes:
- x = Undefined at reset.

FTFL_FEPROT field descriptions

Field	Description
EPROT	EEPROM Region Protect Individual EEPROM regions can be protected from alteration by setting the associated EPROT bit. The EPROT bits are not used when the FlexNVM Partition Code is set to data flash only. When the FlexNVM Partition Code is set to data flash and EEPROM or EEPROM only, each EPROT bit covers one-eighth of the configured EEPROM data (see the EEPROM Data Set Size parameter description). **In NVM Normal mode:** The protection can only be increased. This means that currently-unprotected memory can be protected, but currently-protected memory cannot be unprotected. Since unprotected regions are marked with a 1 and protected regions use a 0, only writes changing 1s to 0s are accepted. This 1-to-0 transition check is performed on a bit-by-bit basis. Those FEPROT bits with 1-to-0 transitions are accepted while all bits with 0-to-1 transitions are ignored. **In NVM Special mode :** All bits of the FEPROT register are writable without restriction. Unprotected areas can be protected and protected areas can be unprotected. **Restriction:** Never write to the FEPROT register while a command is running (CCIF=0). **Reset:** During the reset sequence, the FEPROT register is loaded with the contents of the FlexRAM protection byte in the Flash Configuration Field located in program flash. The flash basis for the reset values is signified by X in the register diagram. To change the EEPROM protection that will be loaded during the reset sequence, the sector of program flash that contains the Flash Configuration Field must be unprotected; then the EEPROM protection byte must be erased and reprogrammed.

FTFL_FEPROT field descriptions (continued)

Field	Description
	Trying to alter data by writing to any protected area in the EEPROM results in a protection violation error and sets the FPVIOL bit in the FSTAT register. 0 EEPROM region is protected 1 EEPROM region is not protected

20.3.4.8 Data Flash Protection Register (FTFL_FDPROT)

The FDPROT register defines which data flash regions are protected against program and erase operations. Protected Flash regions cannot have their content changed; that is, these regions cannot be programmed and cannot be erased by any flash command. Unprotected regions can be changed by both program and erase operations.

Address: 1_C780h base + 17h offset = 1_C797h

Bit	7	6	5	4	3	2	1	0
Read Write	DPROT							
Reset	x*	x*	x*	x*	x*	x*	x*	x*

* Notes:
- x = Undefined at reset.

FTFL_FDPROT field descriptions

Field	Description
DPROT	Data Flash Region Protect Individual data flash regions can be protected from program and erase operations by setting the associated DPROT bit. Each DPROT bit protects one-eighth of the partitioned data flash memory space. The granularity of data flash protection cannot be less than the data flash sector size. If an unused DPROT bit is set, the Erase all Blocks command does not execute and the FSTAT[FPVIOL] flag is set. **In NVM Normal mode:** The protection can only be increased, meaning that currently unprotected memory can be protected but currently protected memory cannot be unprotected. Since unprotected regions are marked with a 1 and protected regions use a 0, only writes changing 1s to 0s are accepted. This 1-to-0 transition check is performed on a bit-by-bit basis. Those FDPROT bits with 1-to-0 transitions are accepted while all bits with 0-to-1 transitions are ignored. **In NVM Special mode:** All bits of the FDPROT register are writable without restriction. Unprotected areas can be protected and protected areas can be unprotected. **Restriction:** The user must never write to the FDPROT register while a command is running (CCIF=0). **Reset:** During the reset sequence, the FDPROT register is loaded with the contents of the data flash protection byte in the Flash Configuration Field located in program flash memory. The flash basis for the reset values is signified by X in the register diagram. To change the data flash protection that will be loaded during the reset sequence, unprotect the sector of program flash that contains the Flash Configuration Field. Then, erase and reprogram the data flash protection byte.

FTFL_FDPROT field descriptions (continued)

Field	Description
	Trying to alter data with the program and erase commands in any protected area in the data flash memory results in a protection violation error and sets the FSTAT[FPVIOL] bit. A full block erase of the data flash memory (see the Erase Flash Block command description) is not possible if the data flash memory contains any protected region or if the FlexNVM block has been partitioned for EEPROM. 0 Data Flash region is protected 1 Data Flash region is not protected

20.4 Functional Description

The information found here describes functional details of the flash memory module.

20.4.1 Flash Protection

Individual regions within the flash memory can be protected from program and erase operations.

Protection is controlled by the following registers:

- FPROT*n* —
  - For 2^n program flash sizes, four registers typically protect 32 regions of the program flash memory as shown in the following figure

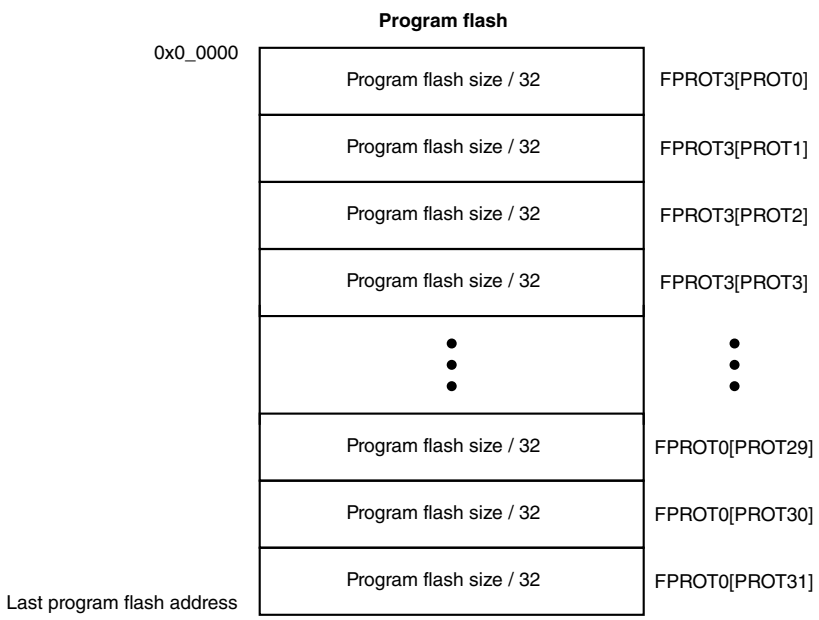


Figure 20-2. Program flash protection

- FDPROT —

- protects eight regions of the data flash memory as shown in the following figure

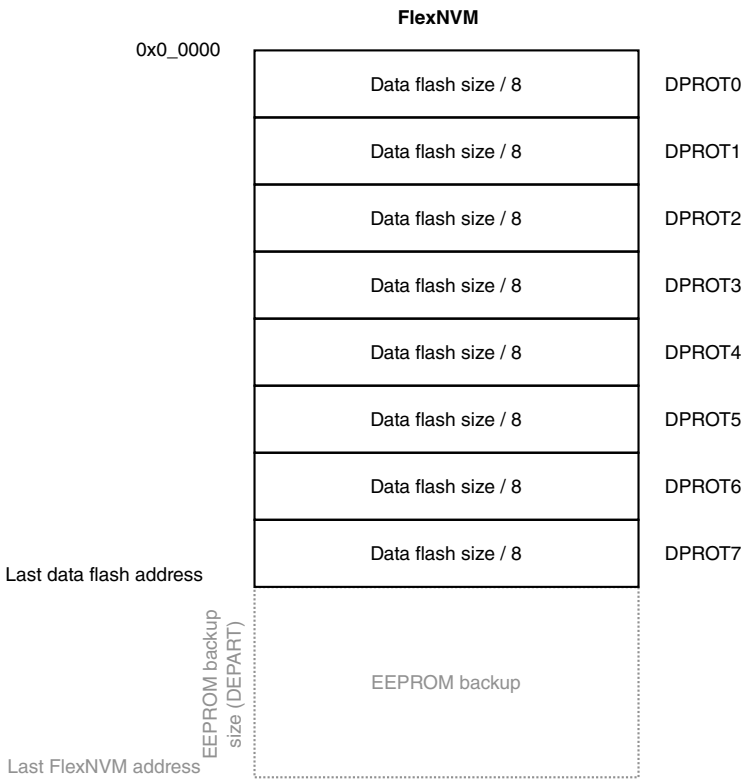


Figure 20-3. Data flash protection

- FEPROT — Protects eight regions of the EEPROM memory as shown in the following figure

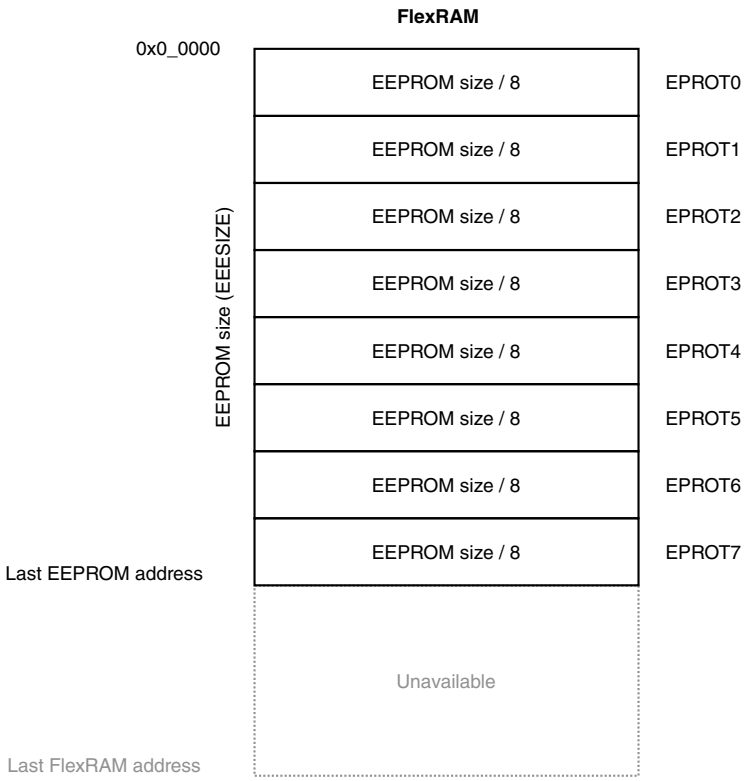


Figure 20-4. EEPROM protection

NOTE

Flash protection features are discussed further in AN4507: Using the Kinetis Security and Flash Protection Features . Not all features described in the application note are available on this device.

20.4.2 FlexNVM Description

This section describes the FlexNVM memory.

20.4.2.1 FlexNVM Block Partitioning for FlexRAM

The user can configure the FlexNVM block as either:

- Basic data flash,
- EEPROM flash records to support the built-in EEPROM feature, or
- A combination of both.

The user's FlexNVM configuration choice is specified using the Program Partition command described in Program Partition Command.

CAUTION

While different partitions of the FlexNVM block are available, the intention is that a single partition choice is used throughout the entire lifetime of a given application. The FlexNVM partition code choices affect the endurance and data retention characteristics of the device.

20.4.2.2 EEPROM User Perspective

The EEPROM system is shown in the following figure.

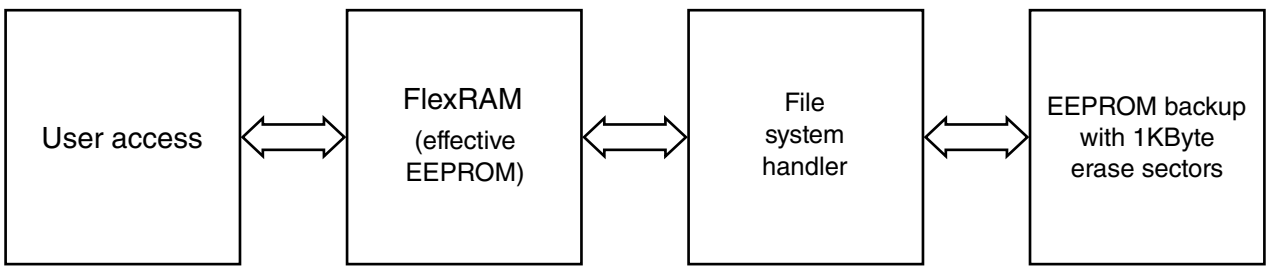


Figure 20-5. Top Level EEPROM Architecture

To handle varying customer requirements, the FlexRAM and FlexNVM blocks can be split into partitions as shown in the figure below.

1. **EEPROM partition** (EEESIZE) — The amount of FlexRAM used for EEPROM can be set from 0 Bytes (no EEPROM) to the maximum FlexRAM size (see Table 20-2). The remainder of the FlexRAM is not accessible while the FlexRAM is configured for EEPROM (see Set FlexRAM Function Command). The EEPROM partition grows upward from the bottom of the FlexRAM address space.
2. **Data flash partition** (DEPART) — The amount of FlexNVM memory used for data flash can be programmed from 0 bytes (all of the FlexNVM block is available for EEPROM backup) to the maximum size of the FlexNVM block (see Table 20-4).
3. **FlexNVM EEPROM partition** — The amount of FlexNVM memory used for EEPROM backup, which is equal to the FlexNVM block size minus the data flash memory partition size. The EEPROM backup size must be at least 16 times the EEPROM partition size in FlexRAM.

The partition information (EEESIZE, DEPART) is stored in the data flash IFR and is programmed using the Program Partition command (see Program Partition Command). Typically, the Program Partition command is executed only once in the lifetime of the device.

Data flash memory is useful for applications that need to quickly store large amounts of data or store data that is static. The EEPROM partition in FlexRAM is useful for storing smaller amounts of data that will be changed often.

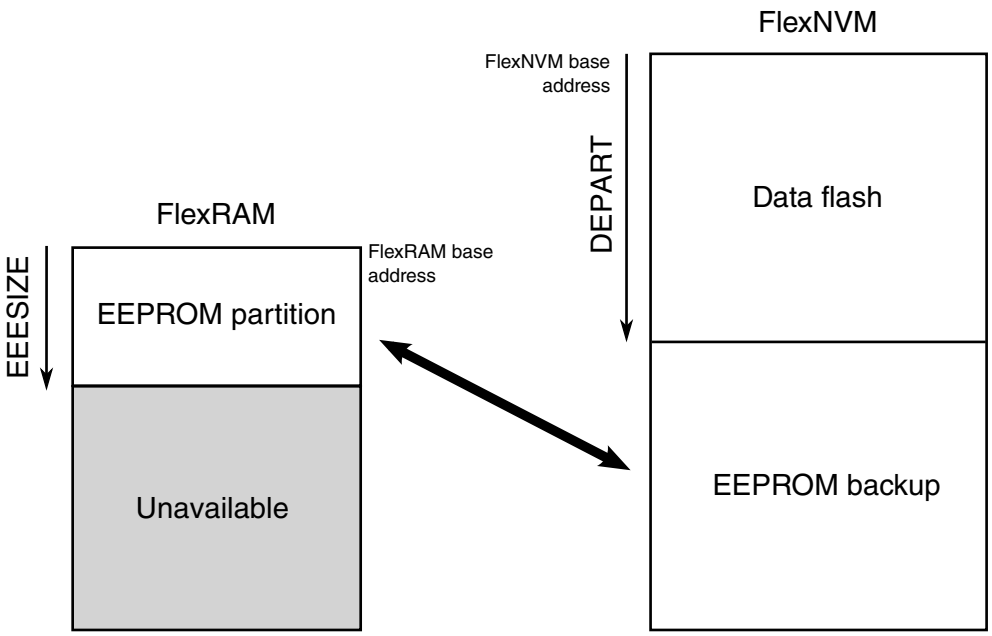


Figure 20-6. FlexRAM to FlexNVM Memory Mapping

20.4.2.3 EEPROM Implementation Overview

Out of reset with the FSTAT[CCIF] bit clear, the partition settings (EEESIZE, DEPART) are read from the data flash IFR and the EEPROM file system is initialized accordingly. The EEPROM file system locates all valid EEPROM data records in EEPROM backup and copies the newest data to FlexRAM. The FSTAT[CCIF] and FCNFG[EEERDY] bits are set after data from all valid EEPROM data records is copied to the FlexRAM. After the CCIF bit is set, the FlexRAM is available for read or write access.

When configured for EEPROM use, writes to an unprotected location in FlexRAM invokes the EEPROM file system to program a new EEPROM data record in the EEPROM backup memory in a round-robin fashion. As needed, the EEPROM file system identifies the EEPROM backup sector that is being erased for future use and partially erases that EEPROM backup sector. After a write to the FlexRAM, the FlexRAM is not accessible until the FSTAT[CCIF] bit is set. The FCNFG[EEERDY] bit will also be set. If enabled, the interrupt associated with the FSTAT[CCIF] bit can be used to determine when the FlexRAM is available for read or write access.

After a sector in EEPROM backup is full of EEPROM data records, EEPROM data records from the sector holding the oldest data are gradually copied over to a previously-erased EEPROM backup sector. When the sector copy completes, the EEPROM backup sector holding the oldest data is tagged for erase.

20.4.2.4 Write endurance to FlexRAM for EEPROM

When the FlexNVM partition code is not set to full data flash, the EEPROM data set size can be set to any of several non-zero values.

The bytes not assigned to data flash via the FlexNVM partition code are used by the flash memory module to obtain an effective endurance increase for the EEPROM data. The built-in EEPROM record management system raises the number of program/erase cycles that can be attained prior to device wear-out by cycling the EEPROM data through a larger EEPROM NVM storage space.

While different partitions of the FlexNVM are available, the intention is that a single choice for the FlexNVM partition code and EEPROM data set size is used throughout the entire lifetime of a given application. The EEPROM endurance equation and graph shown below assume that only one configuration is ever used.

$$\text{Writes_FlexRAM} = \frac{\text{EEPROM} - 2 \times \text{EEESIZE}}{\text{EEESIZE}} \times \text{Write_efficiency} \times n_{\text{nvmcycd}}$$

where

- Writes_FlexRAM — minimum number of writes to each FlexRAM location
- EEPROM — allocated FlexNVM based on DEPART; entered with the Program Partition command
- EEESIZE — allocated FlexRAM based on DEPART; entered with the Program Partition command
- Write_efficiency —
  - 0.25 for 8-bit writes to FlexRAM
  - 0.50 for 16-bit or 32-bit writes to FlexRAM
- $n_{nvmcycd}$ — data flash cycling endurance (the following graph assumes 10,000 cycles)

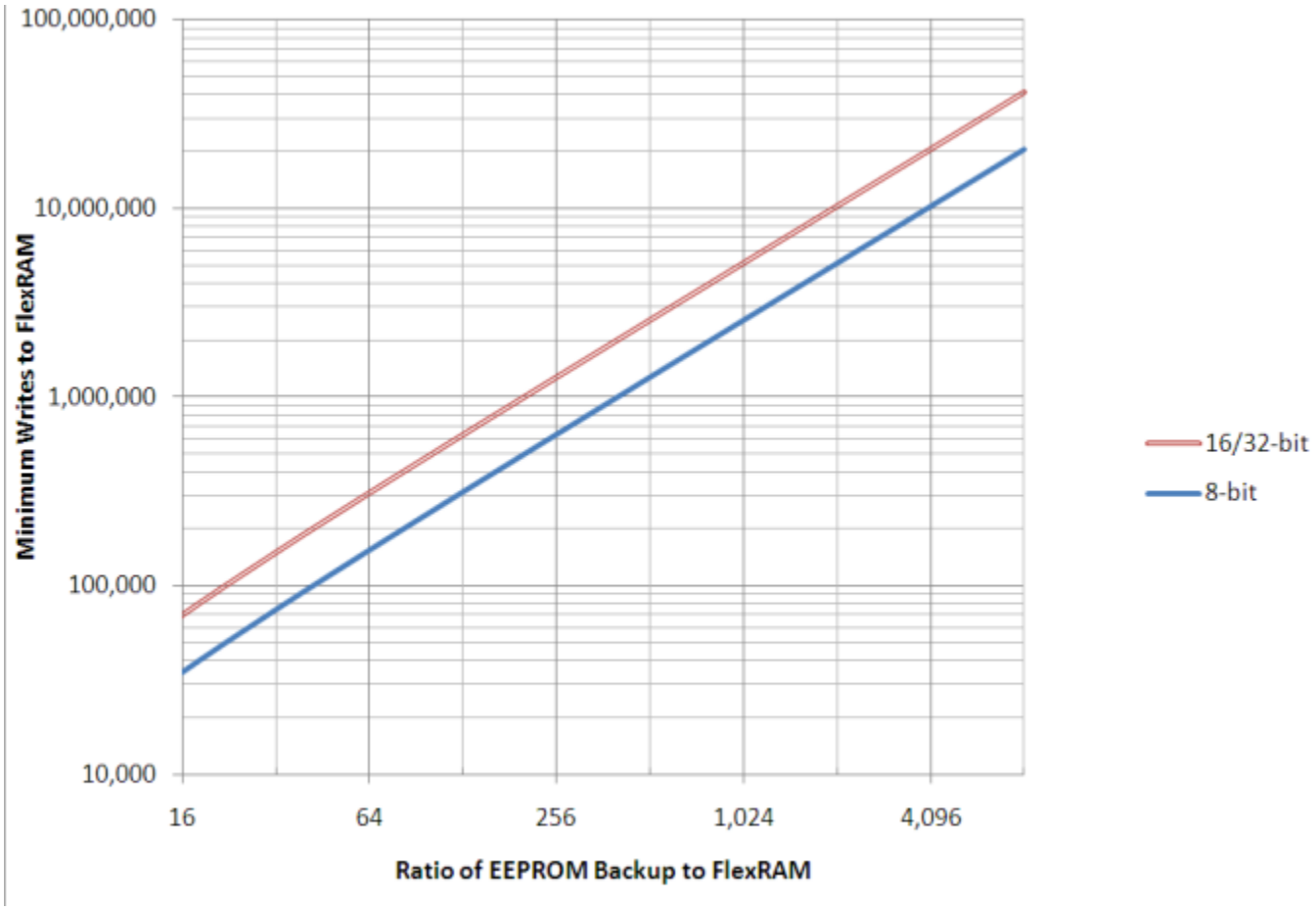


Figure 20-7. EEPROM backup writes to FlexRAM

20.4.3 Interrupts

The flash memory module can generate interrupt requests to the MCU upon the occurrence of various flash events.

These interrupt events and their associated status and control bits are shown in the following table.

Table 20-5. Flash Interrupt Sources

Flash Event	Readable Status Bit	Interrupt Enable Bit
Flash Command Complete	FSTAT[CCIF]	FCNFG[CCIE]
Flash Read Collision Error	FSTAT[RDCOLERR]	FCNFG[RDCOLLIE]

Note

Vector addresses and their relative interrupt priority are determined at the MCU level.

Some devices also generate a bus error response as a result of a Read Collision Error event. See the chip configuration information to determine if a bus error response is also supported.

20.4.4 Flash Operation in Low-Power Modes

20.4.4.1 Wait Mode

When the MCU enters wait mode, the flash memory module is not affected. The flash memory module can recover the MCU from wait via the command complete interrupt (see Interrupts).

20.4.4.2 Stop Mode

When the MCU requests stop mode, if a flash command is active (CCIF = 0) the command execution completes before the MCU is allowed to enter stop mode.

CAUTION

The MCU should never enter stop mode while any flash command is running (CCIF = 0).

NOTE

While the MCU is in very-low-power modes (VLPR, VLPW, VLPS), the flash memory module does not accept flash commands.

20.4.5 Functional Modes of Operation

The flash memory module has two operating modes: NVM Normal and NVM Special.

The operating mode affects the command set availability (see Table 20-6). Refer to the Chip Configuration details of this device for how to activate each mode.

20.4.6 Flash Reads and Ignored Writes

The flash memory module requires only the flash address to execute a flash memory read.

The MCU must not read from the flash memory while commands are running (as evidenced by CCIF=0) on that block. Read data cannot be guaranteed from a flash block while any command is processing within that block. The block arbitration logic detects any simultaneous access and reports this as a read collision error (see the FSTAT[RDCOLERR] bit).

20.4.7 Read While Write (RWW)

The following simultaneous accesses are allowed:

- The user may read from the program flash memory while commands (typically program and erase operations) are active in the data flash and FlexRAM memory space.
- The MCU can fetch instructions from program flash during both data flash program and erase operations and while EEPROM backup data is maintained by the EEPROM commands.
- Conversely, the user may read from data flash and FlexRAM while program and erase commands are executing on the program flash.
- When configured as traditional RAM, writes to the FlexRAM are allowed during program and data flash operations.

Simultaneous data flash operations and FlexRAM writes, when FlexRAM is used for EEPROM, are not possible.

Simultaneous operations are further discussed in Allowed Simultaneous Flash Operations.

20.4.8 Flash Program and Erase

All flash functions except read require the user to setup and launch a flash command through a series of peripheral bus writes.

The user cannot initiate any further flash commands until notified that the current command has completed. The flash command structure and operation are detailed in Flash Command Operations.

20.4.9 Flash Command Operations

Flash command operations are typically used to modify flash memory contents.

The next sections describe:

- The command write sequence used to set flash command parameters and launch execution
- A description of all flash commands available

20.4.9.1 Command Write Sequence

Flash commands are specified using a command write sequence illustrated in Figure 20-8. The flash memory module performs various checks on the command (FCCOB) content and continues with command execution if all requirements are fulfilled.

Before launching a command, the ACCERR and FPVIOL bits in the FSTAT register must be zero and the CCIF flag must read 1 to verify that any previous command has completed. If CCIF is zero, the previous command execution is still active, a new command write sequence cannot be started, and all writes to the FCCOB registers are ignored.

Attempts to launch a flash command in VLP mode will be ignored.

20.4.9.1.1 Load the FCCOB Registers

The user must load the FCCOB registers with all parameters required by the desired flash command. The individual registers that make up the FCCOB data set can be written in any order.

20.4.9.1.2 Launch the Command by Clearing CCIF

Once all relevant command parameters have been loaded, the user launches the command by clearing FSTAT[CCIF] by writing a '1' to it. FSTAT[CCIF] remains 0 until the flash command completes.

The FSTAT register contains a blocking mechanism that prevents a new command from launching (can't clear FSTAT[CCIF]) if the previous command resulted in an access error (FSTAT[ACCERR]=1) or a protection violation (FSTAT[FPVIOL]=1). In error scenarios, two writes to FSTAT are required to initiate the next command: the first write clears the error flags, the second write clears CCIF.

20.4.9.1.3 Command Execution and Error Reporting

The command processing has several steps:

1. The flash memory module reads the command code and performs a series of parameter checks and protection checks, if applicable, which are unique to each command.

   If the parameter check fails, the FSTAT[ACCERR] (access error) flag is set. FSTAT[ACCERR] reports invalid instruction codes and out-of bounds addresses. Usually, access errors suggest that the command was not set-up with valid parameters in the FCCOB register group.

   Program and erase commands also check the address to determine if the operation is requested to execute on protected areas. If the protection check fails, FSTAT[FPVIOL] (protection error) flag is set.

   Command processing never proceeds to execution when the parameter or protection step fails. Instead, command processing is terminated after setting FSTAT[CCIF].

2. If the parameter and protection checks pass, the command proceeds to execution. Run-time errors, such as failure to erase verify, may occur during the execution phase. Run-time errors are reported in FSTAT[MGSTAT0]. A command may have access errors, protection errors, and run-time errors, but the run-time errors are not seen until all access and protection errors have been corrected.
3. Command execution results, if applicable, are reported back to the user via the FCCOB and FSTAT registers.
4. The flash memory module sets FSTAT[CCIF] signifying that the command has completed.

The flow for a generic command write sequence is illustrated in the following figure.

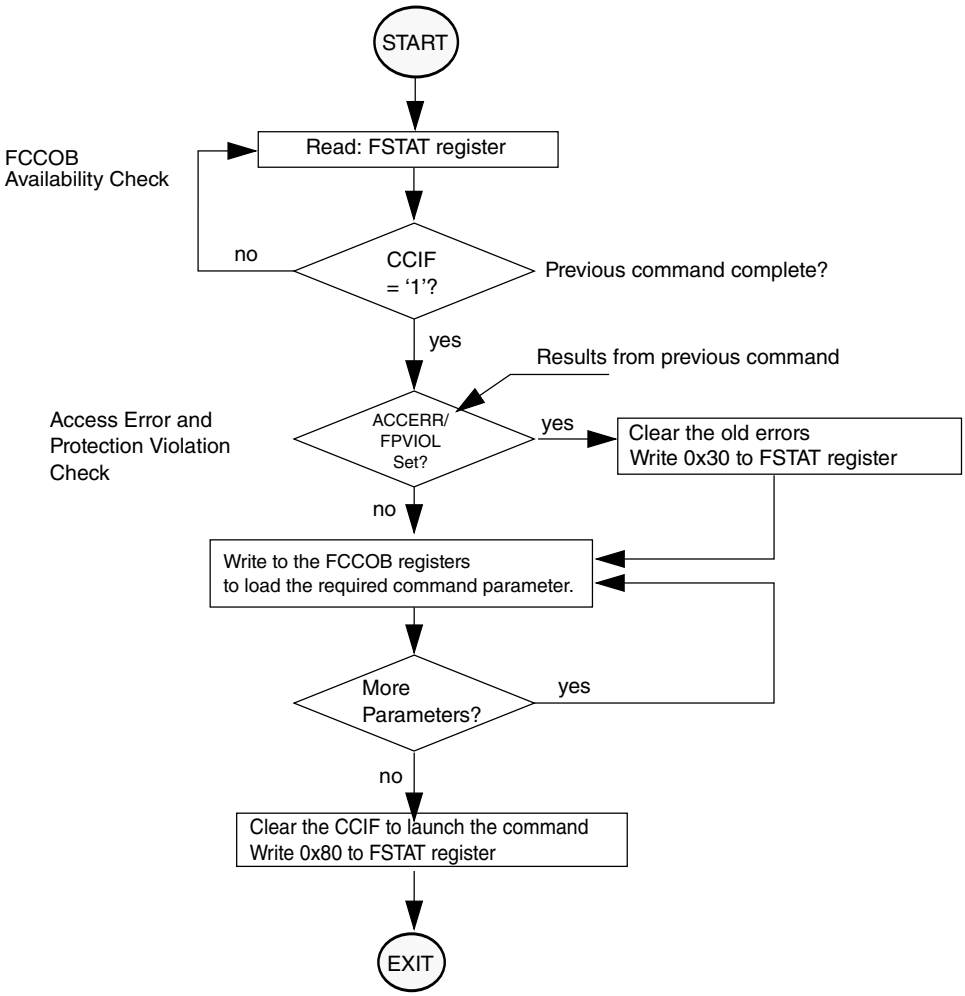


Figure 20-8. Generic flash command write sequence flowchart

20.4.9.2 Flash Commands

The following table summarizes the function of all flash commands. If the program flash, data flash, or FlexRAM column is marked with an 'X', the flash command is relevant to that particular memory resource.

FCMD	Command	Program flash	Data flash	FlexRAM	Function
0x00	Read 1s Block	×	×		Verify that a program flash or data flash block is erased. FlexNVM block must not be partitioned for EEPROM.
0x01	Read 1s Section	×	×		Verify that a given number of program flash or data flash

Table continues on the next page...

FCMD	Command	Program flash	Data flash	FlexRAM	Function
					locations from a starting address are erased.
0x02	Program Check	×	×		Tests previously-programmed locations at margin read levels.
0x03	Read Resource	IFR, ID	IFR		Read 4 bytes from program flash IFR, data flash IFR, or version ID.
0x06	Program Longword	×	×		Program 4 bytes in a program flash block or a data flash block.
0x08	Erase Flash Block	×	×		Erase a program flash block or data flash block. An erase of any flash block is only possible when unprotected. FlexNVM block must not be partitioned for EEPROM.
0x09	Erase Flash Sector	×	×		Erase all bytes in a program flash or data flash sector.
0x0B	Program Section	×	×	×	Program data from the Section Program Buffer to a program flash or data flash block.
0x40	Read 1s All Blocks	×	×		Verify that all program flash, data flash blocks, EEPROM backup data records, and data flash IFR are erased then release MCU security.
0x41	Read Once	IFR			Read 4 bytes of a dedicated 64 byte field in the program flash IFR.
0x43	Program Once	IFR			One-time program of 4 bytes of a dedicated 64-byte field in the program flash IFR.

Table continues on the next page...

FCMD	Command	Program flash	Data flash	FlexRAM	Function
0x44	Erase All Blocks	×	×	×	Erase all program flash blocks, data flash blocks, FlexRAM, EEPROM backup data records, and data flash IFR. Then, verify-erase and release MCU security. **NOTE:** An erase is only possible when all memory locations are unprotected.
0x45	Verify Backdoor Access Key	×			Release MCU security after comparing a set of user-supplied security keys to those stored in the program flash.
0x80	Program Partition		IFR	×	Program the FlexNVM Partition Code and EEPROM Data Set Size into the data flash IFR. Format all EEPROM backup data sectors allocated for EEPROM. Initialize the FlexRAM.
0x81	Set FlexRAM Function		x	×	Switches FlexRAM function between RAM and EEPROM. When switching to EEPROM, FlexNVM is not available while valid data records are being copied from EEPROM backup to FlexRAM.

NOTE

FlexRAM, or Programming Acceleration RAM, is used during PGMSEC command.

20.4.9.3 Flash Commands by Mode

The following table shows the flash commands that can be executed in each flash operating mode.

Table 20-6. Flash Commands by Mode

FCMD	Command	NVM Normal			NVM Special		
		Unsecure	Secure	MEEN=10	Unsecure	Secure	MEEN=10
0x00	Read 1s Block	×	×	×	×	—	—
0x01	Read 1s Section	×	×	×	×	—	—
0x02	Program Check	×	×	×	×	—	—
0x03	Read Resource	×	×	×	×	—	—
0x06	Program Longword	×	×	×	×	—	—
0x08	Erase Flash Block	×	×	×	×	—	—
0x09	Erase Flash Sector	×	×	×	×	—	—
0x0B	Program Section	×	×	×	×	—	—
0x40	Read 1s All Blocks	×	×	×	×	×	—
0x41	Read Once	×	×	×	×	—	—
0x43	Program Once	×	×	×	×	—	—
0x44	Erase All Blocks	×	×	×	×	×	—
0x45	Verify Backdoor Access Key	×	×	×	×	—	—
0x80	Program Partition	×	×	×	×	—	—
0x81	Set FlexRAM Function	×	×	×	×	—	—

20.4.9.4 Allowed Simultaneous Flash Operations

Only the operations marked 'OK' in the following table are permitted to run simultaneously on the program flash, data flash, and FlexRAM memories. Some operations cannot be executed simultaneously because certain hardware resources are shared by the memories. The priority has been placed on permitting program flash reads while program and erase operations execute on the FlexNVM and FlexRAM. This provides read (program flash) while write (FlexNVM, FlexRAM) functionality.

Table 20-7. Allowed Simultaneous Memory Operations

		Program Flash			Data Flash			FlexRAM		
		Read	Program	Sector Erase	Read	Program	Sector Erase	Read	E-Write	R-Write
Program flash	Read	—				OK	OK		OK	
	Program		—		OK			OK		OK[3]
	Sector Erase			—	OK			OK		OK
Data flash	Read		OK	OK	—					
	Program	OK				—		OK		OK
	Sector Erase	OK					—	OK		OK
FlexRAM	Read		OK	OK		OK	OK	—		
	E-Write[1]	OK							—	
	R-Write[2]		OK	OK		OK	OK			—

1. When FlexRAM configured for EEPROM (writes are effectively multi-cycle operations).
2. When FlexRAM configured as traditional RAM (writes are single-cycle operations).
3. When FlexRAM configured as traditional RAM, writes to the RAM are ignored while the Program Section command is active (CCIF = 0).

20.4.10 Margin Read Commands

The Read-1s commands (Read 1s All Blocks, Read 1s Block, and Read 1s Section) and the Program Check command have a margin choice parameter that allows the user to apply non-standard read reference levels to the program flash and data flash array reads performed by these commands.

Using the preset 'user' and 'factory' margin levels, these commands perform their associated read operations at tighter tolerances than a 'normal' read. These non-standard read levels are applied only during the command execution. All simple (uncommanded) flash array reads to the MCU always use the standard, un-margined, read reference level.

Only the 'normal' read level should be employed during normal flash usage. The non-standard, 'user' and 'factory' margin levels should be employed only in special cases. They can be used during special diagnostic routines to gain confidence that the device is not suffering from the end-of-life data loss customary of flash memory devices.

Erased ('1') and programmed ('0') bit states can degrade due to elapsed time and data cycling (number of times a bit is erased and re-programmed). The lifetime of the erased states is relative to the last erase operation. The lifetime of the programmed states is measured from the last program time.

The 'user' and 'factory' levels become, in effect, a minimum safety margin; i.e. if the reads pass at the tighter tolerances of the 'user' and 'factory' margins, then the 'normal' reads have at least this much safety margin before they experience data loss.

The 'user' margin is a small delta to the normal read reference level. 'User' margin levels can be employed to check that flash memory contents have adequate margin for normal level read operations. If unexpected read results are encountered when checking flash memory contents at the 'user' margin levels, loss of information might soon occur during 'normal' readout.

The 'factory' margin is a bigger deviation from the norm, a more stringent read criteria that should only be attempted immediately (or very soon) after completion of an erase or program command, early in the cycling life. 'Factory' margin levels can be used to check that flash memory contents have adequate margin for long-term data retention at the normal level setting. If unexpected results are encountered when checking flash memory contents at 'factory' margin levels, the flash memory contents should be erased and reprogrammed.

CAUTION

Factory margin levels must only be used during verify of the initial factory programming.

20.4.11 Flash Command Description

This section describes all flash commands that can be launched by a command write sequence.

The flash memory module sets the FSTAT[ACCERR] bit and aborts the command execution if any of the following illegal conditions occur:

- There is an unrecognized command code in the FCCOB FCMD field.
- There is an error in a FCCOB field for the specific commands. Refer to the error handling table provided for each command.

Ensure that FSTAT[ACCERR] and FSTAT[FPVIOL] are cleared prior to starting the command write sequence. As described in Launch the Command by Clearing CCIF, a new command cannot be launched while these error flags are set.

Do not attempt to read a flash block while the flash memory module is running a command (FSTAT[CCIF] = 0) on that same block. The flash memory module may return invalid data to the MCU with the collision error flag (FSTAT[RDCOLERR]) set.

When required by the command, address bit 23 selects between:

- program flash (=0)
- data flash (=1)

CAUTION

Flash data must be in the erased state before being programmed. Cumulative programming of bits (adding more zeros) is not allowed.

20.4.11.1 Read 1s Block Command

The Read 1s Block command checks to see if an entire program flash or data flash block has been erased to the specified margin level. The FCCOB flash address bits determine which flash block is erase-verified.

Table 20-8. Read 1s Block Command FCCOB Requirements

FCCOB Number	FCCOB Contents [7:0]
0	0x00 (RD1BLK)
1	Flash address [23:16] in the flash block to be verified
2	Flash address [15:8] in the flash block to be verified
3	Flash address [7:0][1] in the flash block to be verified
4	Read-1 Margin Choice

1. Must be longword aligned (Flash address [1:0] = 00).

After clearing CCIF to launch the Read 1s Block command, the flash memory module sets the read margin for 1s according to Table 20-9 and then reads all locations within the selected program flash or data flash block.

When the data flash is targeted, DEPART must be set for no EEPROM, else the Read 1s Block command aborts setting the FSTAT[ACCERR] bit. If the flash memory module fails to read all 1s (i.e. the flash block is not fully erased), the FSTAT[MGSTAT0] bit is set. The CCIF flag sets after the Read 1s Block operation has completed.

Table 20-9. Margin Level Choices for Read 1s Block

Read Margin Choice	Margin Level Description
0x00	Use the 'normal' read level for 1s
0x01	Apply the 'User' margin to the normal read-1 level
0x02	Apply the 'Factory' margin to the normal read-1 level

Table 20-10. Read 1s Block Command Error Handling

Error Condition	Error Bit
Command not available in current mode/security	FSTAT[ACCERR]
An invalid margin choice is specified	FSTAT[ACCERR]
Program flash is selected and the address is out of program flash range	FSTAT[ACCERR]
Data flash is selected and the address is out of data flash range	FSTAT[ACCERR]
Data flash is selected with EEPROM enabled	FSTAT[ACCERR]
Flash address is not longword aligned	FSTAT[ACCERR]
Read-1s fails	FSTAT[MGSTAT0]

20.4.11.2 Read 1s Section Command

The Read 1s Section command checks if a section of program flash or data flash memory is erased to the specified read margin level. The Read 1s Section command defines the starting address and the number of phrases or longwords to be verified.

Table 20-11. Read 1s Section Command FCCOB Requirements (P-Flash)

FCCOB Number	FCCOB Contents [7:0]
0	0x01 (RD1SEC)
1	Flash address [23:16] of the first phrase to be verified
2	Flash address [15:8] of the first phrase to be verified
3	Flash address [7:0][1] of the first phrase to be verified
4	Number of phrases to be verified [15:8]
5	Number of phrases to be verified [7:0]
6	Read-1 Margin Choice

1. Must be phrase aligned (Flash address [2:0] = 000).

Table 20-12. Read 1s Section Command FCCOB Requirements (D-Flash)

FCCOB Number	FCCOB Contents [7:0]
0	0x01 (RD1SEC)
1	Flash address [23:16] of the first longword to be verified
2	Flash address [15:8] of the first longword to be verified
3	Flash address [7:0][1] of the first longword to be verified
4	Number of longwords to be verified [15:8]
5	Number of longwords to be verified [7:0]
6	Read-1 Margin Choice

1. Must be longword aligned (Flash address [1:0] = 00).

Upon clearing CCIF to launch the Read 1s Section command, the flash memory module sets the read margin for 1s according to Table 20-13 and then reads all locations within the specified section of flash memory. If the flash memory module fails to read all 1s (that is, the flash section is not erased), FSTAT[MGSTAT0] is set. FSTAT[CCIF] sets after the Read 1s Section operation completes.

Table 20-13. Margin Level Choices for Read 1s Section

Read Margin Choice	Margin Level Description
0x00	Use the 'normal' read level for 1s
0x01	Apply the 'User' margin to the normal read-1 level
0x02	Apply the 'Factory' margin to the normal read-1 level

Table 20-14. Read 1s Section Command Error Handling

Error condition	Error bit
Command not available in current mode/security	FSTAT[ACCERR]
An invalid margin code is supplied.	FSTAT[ACCERR]
An invalid flash address is supplied.	FSTAT[ACCERR]
Flash address is not phrase/longword aligned.	FSTAT[ACCERR]
The requested section crosses a Flash block boundary.	FSTAT[ACCERR]
The requested number of phrases/longwords is 0.	FSTAT[ACCERR]
Read-1s fails.	FSTAT[MGSTAT0]

20.4.11.3 Program Check Command

The Program Check command tests a previously programmed program flash or data flash longword to see if it reads correctly at the specified margin level.

Table 20-15. Program Check Command FCCOB Requirements

FCCOB Number	FCCOB Contents [7:0]
0	0x02 (PGMCHK)
1	Flash address [23:16]
2	Flash address [15:8]
3	Flash address [7:0][1]
4	Margin Choice
8	Byte 0 expected data
9	Byte 1 expected data
A	Byte 2 expected data
B	Byte 3 expected data

1. Must be longword aligned (Flash address [1:0] = 00).

Upon clearing CCIF to launch the Program Check command, the flash memory module sets the read margin for 1s according to Table 20-16, reads the specified longword, and compares the actual read data to the expected data provided by the FCCOB. If the comparison at margin-1 fails, FSTAT[MGSTAT0] is set.

The flash memory module then sets the read margin for 0s, re-reads, and compares again. If the comparison at margin-0 fails, FSTAT[MGSTAT0] is set. FSTAT[CCIF] is set after the Program Check operation completes.

The supplied address must be longword aligned (the lowest two bits of the byte address must be 00):

- Byte 3 data is written to the supplied byte address ('start'),
- Byte 2 data is programmed to byte address start+0b01,
- Byte 1 data is programmed to byte address start+0b10,
- Byte 0 data is programmed to byte address start+0b11.

NOTE

See the description of margin reads, Margin Read Commands

Table 20-16. Margin Level Choices for Program Check

Read Margin Choice	Margin Level Description
0x01	Read at 'User' margin-1 and 'User' margin-0
0x02	Read at 'Factory' margin-1 and 'Factory' margin-0

Table 20-17. Program Check Command Error Handling

Error Condition	Error Bit
Command not available in current mode/security	FSTAT[ACCERR]
An invalid flash address is supplied	FSTAT[ACCERR]
Flash address is not longword aligned	FSTAT[ACCERR]
An invalid margin choice is supplied	FSTAT[ACCERR]
Either of the margin reads does not match the expected data	FSTAT[MGSTAT0]

20.4.11.4 Read Resource Command

The Read Resource command allows the user to read data from special-purpose memory resources located within the flash memory module. The special-purpose memory resources available include program flash IFR space, data flash IFR space, and the Version ID field. Each resource is assigned a select code as shown in Table 20-19.

Table 20-18. Read Resource Command FCCOB Requirements

FCCOB Number	FCCOB Contents [7:0]
0	0x03 (RDRSRC)
1	Flash address [23:16]
2	Flash address [15:8]
3	Flash address [7:0][1]
Returned Values	
4	Read Data [31:24]
5	Read Data [23:16]
6	Read Data [15:8]
7	Read Data [7:0]
User-provided values	
8	Resource Select Code (see Table 20-19)

1. Must be longword aligned (Flash address [1:0] = 00).

Table 20-19. Read Resource Select Codes

Resource Select Code	Description	Resource Size	Local Address Range
0x00	Program Flash 0 IFR	256 Bytes	0x00_0000–0x00_00FF
0x00	Data Flash 0 IFR	256 Bytes	0x80_0000 - 0x80_00FF
0x01[1]	Version ID	8 Bytes	0x00_0000–0x00_0007

1. Located in program flash 0 reserved space.

After clearing CCIF to launch the Read Resource command, four consecutive bytes are read from the selected resource at the provided relative address and stored in the FCCOB register. The CCIF flag sets after the Read Resource operation completes. The Read Resource command exits with an access error if an invalid resource code is provided or if the address for the applicable area is out-of-range.

Table 20-20. Read Resource Command Error Handling

Error Condition	Error Bit
Command not available in current mode/security	FSTAT[ACCERR]
An invalid resource code is entered	FSTAT[ACCERR]
Flash address is out-of-range for the targeted resource.	FSTAT[ACCERR]
Flash address is not longword aligned	FSTAT[ACCERR]

20.4.11.5 Program Longword Command

The Program Longword command programs four previously-erased bytes in the program flash memory or in the data flash memory using an embedded algorithm.

CAUTION

A flash memory location must be in the erased state before being programmed. Cumulative programming of bits (back-to-back program operations without an intervening erase) within a flash memory location is not allowed. Re-programming of existing 0s to 0 is not allowed as this overstresses the device.

Table 20-21. Program Longword Command FCCOB Requirements

FCCOB Number	FCCOB Contents [7:0]
0	0x06 (PGM4)
1	Flash address [23:16]
2	Flash address [15:8]
3	Flash address [7:0][1]
4	Byte 0 program value
5	Byte 1 program value
6	Byte 2 program value
7	Byte 3 program value

1. Must be longword aligned (Flash address [1:0] = 00).

Upon clearing CCIF to launch the Program Longword command, the flash memory module programs the data bytes into the flash using the supplied address. The targeted flash locations must be currently unprotected (see the description of the FPROT and FDPROT registers) to permit execution of the Program Longword operation.

The programming operation is unidirectional. It can only move NVM bits from the erased state ('1') to the programmed state ('0'). Erased bits that fail to program to the '0' state are flagged as errors in FSTAT[MGSTAT0]. The CCIF flag is set after the Program Longword operation completes.

The supplied address must be longword aligned (flash address [1:0] = 00):

- Byte 3 data is written to the supplied byte address ('start'),
- Byte 2 data is programmed to byte address start+0b01,
- Byte 1 data is programmed to byte address start+0b10, and
- Byte 0 data is programmed to byte address start+0b11.

Table 20-22. Program Longword Command Error Handling

Error Condition	Error Bit
Command not available in current mode/security	FSTAT[ACCERR]
An invalid flash address is supplied	FSTAT[ACCERR]
Flash address is not longword aligned	FSTAT[ACCERR]
Flash address points to a protected area	FSTAT[FPVIOL]
Any errors have been encountered during the verify operation	FSTAT[MGSTAT0]

20.4.11.6 Erase Flash Block Command

The Erase Flash Block operation erases all addresses in a single program flash or data flash block.

Table 20-23. Erase Flash Block Command FCCOB Requirements

FCCOB Number	FCCOB Contents [7:0]
0	0x08 (ERSBLK)
1	Flash address [23:16] in the flash block to be erased
2	Flash address [15:8] in the flash block to be erased
3	Flash address [7:0][1] in the flash block to be erased

1. Must be longword aligned (Flash address [1:0] = 00).

Upon clearing CCIF to launch the Erase Flash Block command, the flash memory module erases the main array of the selected flash block and verifies that it is erased. When the data flash is targeted, DEPART must be set for no EEPROM (see Table 20-4) else the Erase Flash Block command aborts setting the FSTAT[ACCERR] bit. The Erase Flash Block command aborts and sets the FSTAT[FPVIOL] bit if any region within the block is protected (see the description of the FPROT and FDPROT registers). If the erase verify fails, FSTAT[MGSTAT0] is set. The CCIF flag will set after the Erase Flash Block operation has completed.

Table 20-24. Erase Flash Block Command Error Handling

Error Condition	Error Bit
Command not available in current mode/security	FSTAT[ACCERR]
Program flash is selected and the address is out of program flash range	FSTAT[ACCERR]
Data flash is selected and the address is out of data flash range	FSTAT[ACCERR]
Data flash is selected with EEPROM enabled	FSTAT[ACCERR]
Flash address is not longword aligned	FSTAT[ACCERR]
Any area of the selected flash block is protected	FSTAT[FPVIOL]
Any errors have been encountered during the verify operation[1]	FSTAT[MGSTAT0]

1. User margin read may be run using the Read 1s Block command to verify all bits are erased.

20.4.11.7 Erase Flash Sector Command

The Erase Flash Sector operation erases all addresses in a flash sector.

Table 20-25. Erase Flash Sector Command FCCOB Requirements

FCCOB Number	FCCOB Contents [7:0]
0	0x09 (ERSSCR)
1	Flash address [23:16] in the flash sector to be erased
2	Flash address [15:8] in the flash sector to be erased
3	Flash address [7:0][1, 1] in the flash sector to be erased

1. For program flash:Must be phrase aligned (flash address [2:0] = 000).
1. For data flash:Must be longword aligned (flash address [1:0] = 00).

After clearing CCIF to launch the Erase Flash Sector command, the flash memory module erases the selected program flash or data flash sector and then verifies that it is erased. The Erase Flash Sector command aborts if the selected sector is protected (see the description of the FPROT and FDPROT registers). If the erase-verify fails the FSTAT[MGSTAT0] bit is set. The CCIF flag is set after the Erase Flash Sector operation completes. The Erase Flash Sector command is suspendable (see the FCNFG[ERSSUSP] bit and Figure 20-9).

Table 20-26. Erase Flash Sector Command Error Handling

Error Condition	Error Bit
Command not available in current mode/security	FSTAT[ACCERR]
An invalid Flash address is supplied	FSTAT[ACCERR]
Flash address is not phrase/longword aligned	FSTAT[ACCERR]
The selected program flash or data flash sector is protected	FSTAT[FPVIOL]
Any errors have been encountered during the verify operation[1]	FSTAT[MGSTAT0]

1. User margin read may be run using the Read 1s Section command to verify all bits are erased.

20.4.11.7.1 Suspending an Erase Flash Sector Operation

To suspend an Erase Flash Sector operation set the FCNFG[ERSSUSP] bit (see Flash Configuration Field Description) when CCIF is clear and the CCOB command field holds the code for the Erase Flash Sector command. During the Erase Flash Sector operation (see Erase Flash Sector Command), the flash memory module samples the state of the ERSSUSP bit at convenient points. If the flash memory module detects that the

ERSSUSP bit is set, the Erase Flash Sector operation is suspended and the flash memory module sets CCIF. While ERSSUSP is set, all writes to flash registers are ignored except for writes to the FSTAT and FCNFG registers.

If an Erase Flash Sector operation effectively completes before the flash memory module detects that a suspend request has been made, the flash memory module clears the ERSSUSP bit prior to setting CCIF. When an Erase Flash Sector operation has been successfully suspended, the flash memory module sets CCIF and leaves the ERSSUSP bit set. While CCIF is set, the ERSSUSP bit can only be cleared to prevent the withdrawal of a suspend request before the flash memory module has acknowledged it.

20.4.11.7.2 Resuming a Suspended Erase Flash Sector Operation

If the ERSSUSP bit is still set when CCIF is cleared to launch the next command, the previous Erase Flash Sector operation resumes. The flash memory module acknowledges the request to resume a suspended operation by clearing the ERSSUSP bit. A new suspend request can then be made by setting ERSSUSP. A single Erase Flash Sector operation can be suspended and resumed multiple times.

There is a minimum elapsed time limit between the request to resume the Erase Flash Sector operation (CCIF is cleared) and the request to suspend the operation again (ERSSUSP is set). This minimum time period is required to ensure that the Erase Flash Sector operation will eventually complete. If the minimum period is continually violated, i.e. the suspend requests come repeatedly and too quickly, no forward progress is made by the Erase Flash Sector algorithm. The resume/suspend sequence runs indefinitely without completing the erase.

20.4.11.7.3 Aborting a Suspended Erase Flash Sector Operation

The user may choose to abort a suspended Erase Flash Sector operation by clearing the ERSSUSP bit prior to clearing CCIF for the next command launch. When a suspended operation is aborted, the flash memory module starts the new command using the new FCCOB contents.

While FCNFG[ERSSUSP] is set, a write to the FlexRAM while FCNFG[EEERDY] is set clears ERSSUSP and aborts the suspended operation. The FlexRAM write operation is executed by the flash memory module.

Note

Aborting the erase leaves the bitcells in an indeterminate, partially-erased state. Data in this sector is not reliable until a new erase command fully completes.

The following figure shows how to suspend and resume the Erase Flash Sector operation.

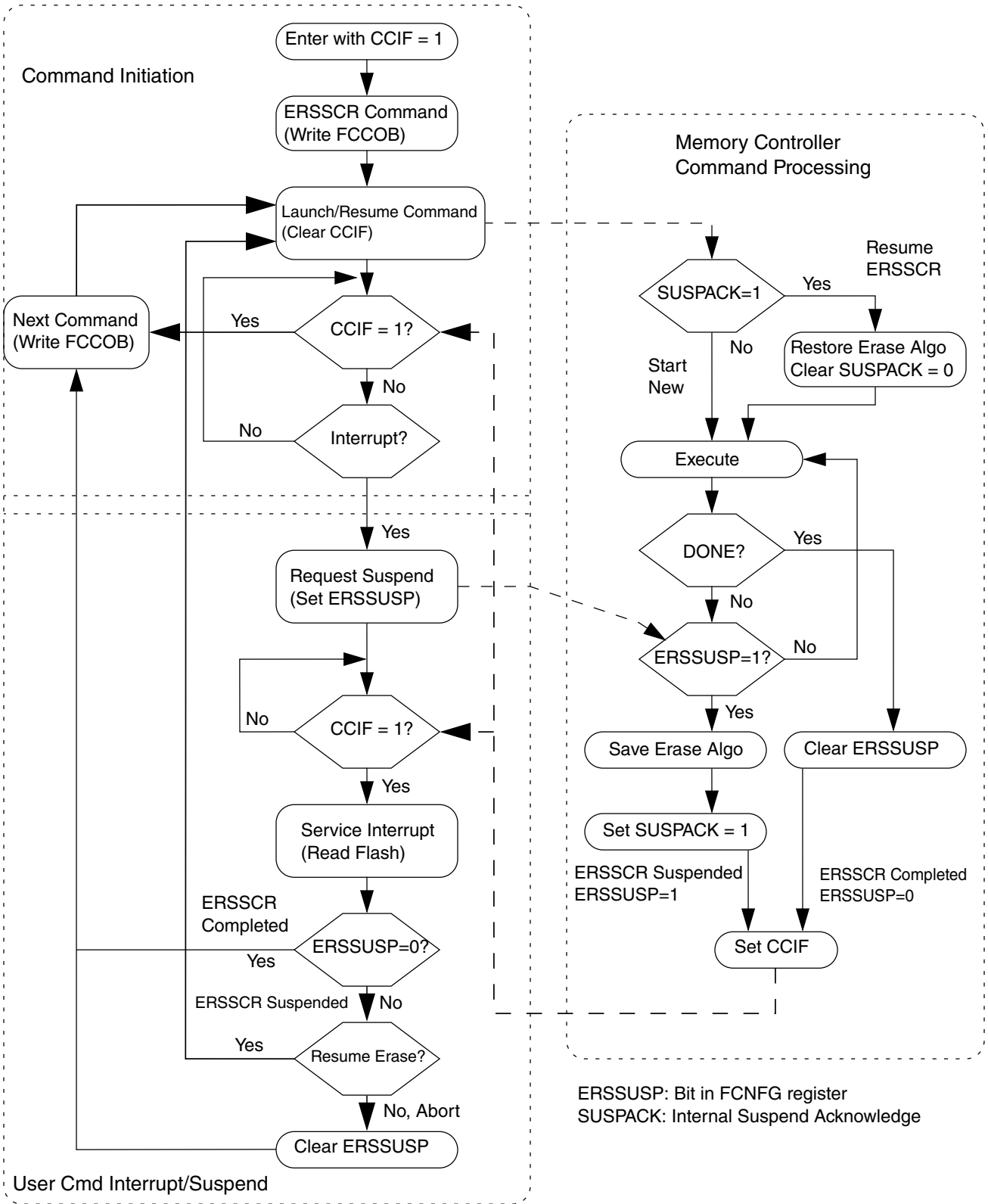


Figure 20-9. Suspend and Resume of Erase Flash Sector Operation

20.4.11.8 Program Section Command

The Program Section operation programs the data found in the section program buffer to previously erased locations in the flash memory using an embedded algorithm. Data is preloaded into the section program buffer by writing to the FlexRAM while it is set to function as traditional RAM or the programming acceleration RAM (see Flash Sector Programming).

The section program buffer is limited to the lower half of the RAM. Data written to the upper half of the RAM is ignored and may be overwritten during Program Section command execution.

CAUTION

A flash memory location must be in the erased state before being programmed. Cumulative programming of bits (back-to-back program operations without an intervening erase) within a flash memory location is not allowed. Re-programming of existing 0s to 0 is not allowed as this overstresses the device.

Table 20-27. Program Section Command FCCOB Requirements (P-Flash)

FCCOB Number	FCCOB Contents [7:0]
0	0x0B (PGMSEC)
1	Flash address [23:16]
2	Flash address [15:8]
3	Flash address [7:0][1]
4	Number of phrases to program [15:8]
5	Number of phrases to program [7:0]

1. Must be phrase aligned (Flash address [2:0] = 000).

Table 20-28. Program Section Command FCCOB Requirements (D-Flash)

FCCOB Number	FCCOB Contents [7:0]
0	0x0B (PGMSEC)
1	Flash address [23:16]
2	Flash address [15:8]
3	Flash address [7:0][1]
4	Number of longwords to program [15:8]
5	Number of longwords to program [7:0]

1. Must be longword aligned (Flash address [1:0] = 00).

After clearing CCIF to launch the Program Section command, the flash memory module blocks access to the FlexRAM and programs the data residing in the section program buffer into the flash memory starting at the flash address provided.

The starting address must be unprotected (see the description of the FPROT and FDPROT registers) to permit execution of the Program Section operation. Programming, which is not allowed to cross a flash sector boundary, continues until all requested phrases or longwords have been programmed. The Program Section command also verifies that after programming, all bits requested to be programmed are programmed.

After the Program Section operation completes, the CCIF flag is set and normal access to the RAM is restored. The contents of the section program buffer may be changed by the Program Section operation.

Table 20-29. Program Section Command Error Handling

Error Condition	Error Bit
Command not available in current mode/security	FSTAT[ACCERR]
An invalid flash address is supplied	FSTAT[ACCERR]
Flash address is not phrase/longword aligned	FSTAT[ACCERR]
The requested section crosses a program flash sector boundary	FSTAT[ACCERR]
The requested number of phrases/longwords is zero	FSTAT[ACCERR]
The space required to store data for the requested number of phrases/longwords is more than half the size of the FlexRAM	FSTAT[ACCERR]
The FlexRAM is not set to function as a traditional RAM, i.e. set if RAMRDY=0	FSTAT[ACCERR]
The flash address falls in a protected area	FSTAT[FPVIOL]
Any errors have been encountered during the verify operation	FSTAT[MGSTAT0]

20.4.11.8.1 Flash Sector Programming

The process of programming an entire flash sector using the Program Section command is as follows:

1. If required, for FlexNVM devices, execute the Set FlexRAM Function command to make the FlexRAM available as traditional RAM and initialize the FlexRAM to all ones.
2. Launch the Erase Flash Sector command to erase the flash sector to be programmed.
3. Beginning with the starting address of the FlexRAM, sequentially write enough data to the RAM to fill an entire flash sector or half the FlexRAM, whichever is less. This area of the RAM serves as the section program buffer.

NOTE

In step 1, the section program buffer was initialized to all ones, the erased state of the flash memory.

The section program buffer can be written to while the operation launched in step 2 is executing, i.e. while CCIF = 0.

4. Execute the Program Section command to program the contents of the section program buffer into the selected flash sector.
5. If a flash sector is larger than half the RAM, repeat steps 3 and 4 until the sector is completely programmed.
6. To program additional flash sectors, repeat steps 2 through 4.
7. To restore EEPROM functionality for FlexNVM devices, execute the Set FlexRAM Function command to make the FlexRAM available as EEPROM.

20.4.11.9 Read 1s All Blocks Command

The Read 1s All Blocks command checks if the program flash blocks, data flash blocks, EEPROM backup records, and data flash IFR have been erased to the specified read margin level, if applicable, and releases security if the readout passes, i.e. all data reads as '1'.

Table 20-30. Read 1s All Blocks Command FCCOB Requirements

FCCOB Number	FCCOB Contents [7:0]
0	0x40 (RD1ALL)
1	Read-1 Margin Choice

After clearing CCIF to launch the Read 1s All Blocks command, the flash memory module :

- sets the read margin for 1s according to Table 20-31,
- checks the contents of the program flash, data flash, EEPROM backup records, and data flash IFR are in the erased state.

If the flash memory module confirms that these memory resources are erased, security is released by setting the FSEC[SEC] field to the unsecure state. The security byte in the flash configuration field (see Flash Configuration Field Description) remains unaffected by the Read 1s All Blocks command. If the read fails, i.e. all memory resources are not in the fully erased state, the FSTAT[MGSTAT0] bit is set.

The EEERDY and RAMRDY bits are clear during the Read 1s All Blocks operation and are restored at the end of the Read 1s All Blocks operation.

The CCIF flag sets after the Read 1s All Blocks operation has completed.

Table 20-31. Margin Level Choices for Read 1s All Blocks

Read Margin Choice	Margin Level Description
0x00	Use the 'normal' read level for 1s
0x01	Apply the 'User' margin to the normal read-1 level
0x02	Apply the 'Factory' margin to the normal read-1 level

Table 20-32. Read 1s All Blocks Command Error Handling

Error Condition	Error Bit
An invalid margin choice is specified	FSTAT[ACCERR]
Read-1s fails	FSTAT[MGSTAT0]

20.4.11.10 Read Once Command

The Read Once command provides read access to a reserved 64-byte field located in the program flash IFR (see Program Flash IFR Map and Program Once Field). Access to the Program Once field is via 16 records, each 4 bytes long. The Program Once field is programmed using the Program Once command described in Program Once Command.

Table 20-33. Read Once Command FCCOB Requirements

FCCOB Number	FCCOB Contents [7:0]
0	0x41 (RDONCE)
1	Program Once record index (0x00 - 0x0F)
2	Not used
3	Not used
Returned Values	
4	Program Once byte 0 value
5	Program Once byte 1 value
6	Program Once byte 2 value
7	Program Once byte 3 value

After clearing CCIF to launch the Read Once command, a 4-byte Program Once record is read from the program flash IFR and stored in the FCCOB register. The CCIF flag is set after the Read Once operation completes. Valid record index values for the Read Once command range from 0x00 to 0x0F. During execution of the Read Once command, any attempt to read addresses within the program flash block containing this 64-byte field returns invalid data. The Read Once command can be executed any number of times.

Table 20-34. Read Once Command Error Handling

Error Condition	Error Bit
Command not available in current mode/security	FSTAT[ACCERR]
An invalid record index is supplied	FSTAT[ACCERR]

20.4.11.11 Program Once Command

The Program Once command enables programming to a reserved 64-byte field in the program flash IFR (see Program Flash IFR Map and Program Once Field). Access to the Program Once field is via 16 records, each 4 bytes long. The Program Once field can be read using the Read Once command (see Read Once Command) or using the Read Resource command (see Read Resource Command). Each Program Once record can be programmed only once since the program flash IFR cannot be erased.

Table 20-35. Program Once Command FCCOB Requirements

FCCOB Number	FCCOB Contents [7:0]
0	0x43 (PGMONCE)
1	Program Once record index (0x00 - 0x0F)
2	Not Used
3	Not Used
4	Program Once byte 0 value
5	Program Once byte 1 value
6	Program Once byte 2 value
7	Program Once byte 3 value

After clearing CCIF to launch the Program Once command, the flash memory module first verifies that the selected record is erased. If erased, then the selected record is programmed using the values provided. The Program Once command also verifies that the programmed values read back correctly. The CCIF flag is set after the Program Once operation has completed.

The reserved program flash IFR location accessed by the Program Once command cannot be erased and any attempt to program one of these records when the existing value is not Fs (erased) is not allowed. Valid record index values for the Program Once command range from 0x00 to 0x0F. During execution of the Program Once command, any attempt to read addresses within the program flash block containing this 64-byte field returns invalid data.

Table 20-36. Program Once Command Error Handling

Error Condition	Error Bit
Command not available in current mode/security	FSTAT[ACCERR]
An invalid record index is supplied	FSTAT[ACCERR]
The requested record has already been programmed to a non-FFFF value[1]	FSTAT[ACCERR]
Any errors have been encountered during the verify operation	FSTAT[MGSTAT0]

1. If a Program Once record is initially programmed to 0xFFFF_FFFF, the Program Once command is allowed to execute again on that same record.

20.4.11.12 Erase All Blocks Command

The Erase All Blocks operation erases all flash memory, initializes the FlexRAM, verifies all memory contents, and releases MCU security.

Table 20-37. Erase All Blocks Command FCCOB Requirements

FCCOB Number	FCCOB Contents [7:0]
0	0x44 (ERSALL)

After clearing CCIF to launch the Erase All Blocks command, the flash memory module erases all program flash memory, data flash memory, data flash IFR space, EEPROM backup memory, and FlexRAM, then verifies that all are erased.

If the flash memory module verifies that all flash memories and the FlexRAM were properly erased, security is released by setting the FSEC[SEC] field to the unsecure state and the FCNFG[RAMRDY] bit is set. The Erase All Blocks command aborts if any flash or FlexRAM region is protected. The security byte and all other contents of the flash configuration field (see Flash Configuration Field Description) are erased by the Erase All Blocks command. If the erase-verify fails, the FSTAT[MGSTAT0] bit is set. The CCIF flag is set after the Erase All Blocks operation completes.

Table 20-38. Erase All Blocks Command Error Handling

Error Condition	Error Bit
Command not available in current mode/security	FSTAT[ACCERR]
Any region of the program flash memory, data flash memory, or FlexRAM is protected	FSTAT[FPVIOL]
Any errors have been encountered during the verify operation[1]	FSTAT[MGSTAT0]

1. User margin read may be run using the Read 1s All Blocks command to verify all bits are erased.

20.4.11.12.1 Triggering an Erase All External to the Flash Memory Module

The functionality of the Erase All BlocksErase All Blocks Unsecure command is also available in an uncommanded fashion outside of the flash memory. Refer to the device's Chip Configuration details for information on this functionality.

Before invoking the external erase all function, the FSTAT[ACCERR and PVIOL] flags must be cleared and the FCCOB0 register must not contain 0x44. When invoked, the erase-all function erases all program flash memory, data flash memory, data flash IFR space, EEPROM backup, and FlexRAM regardless of the protection settings. If the post-erase verify passes, the routine then releases security by setting the FSEC[SEC] field register to the unsecure state and the FCNFG[RAMRDY] bit sets. The security byte in the Flash Configuration Field is also programmed to the unsecure state. The status of the erase-all request is reflected in the FCNFG[ERSAREQ] bit. The FCNFG[ERSAREQ] bit is cleared once the operation completes and the normal FSTAT error reporting is available as described in Erase All Blocks Command.

20.4.11.13 Verify Backdoor Access Key Command

The Verify Backdoor Access Key command only executes if the mode and security conditions are satisfied (see Flash Commands by Mode). Execution of the Verify Backdoor Access Key command is further qualified by the FSEC[KEYEN] bits. The Verify Backdoor Access Key command releases security if user-supplied keys in the FCCOB match those stored in the Backdoor Comparison Key bytes of the Flash Configuration Field (see Flash Configuration Field Description). The column labelled Flash Configuration Field offset address shows the location of the matching byte in the Flash Configuration Field.

Table 20-39. Verify Backdoor Access Key Command FCCOB Requirements

FCCOB Number	FCCOB Contents [7:0]	Flash Configuration Field Offset Address
0	0x45 (VFYKEY)	
1-3	Not Used	
4	Key Byte 0	0x0_0000
5	Key Byte 1	0x0_0001
6	Key Byte 2	0x0_0002
7	Key Byte 3	0x0_0003
8	Key Byte 4	0x0_0004
9	Key Byte 5	0x0_0005
A	Key Byte 6	0x0_0006
B	Key Byte 7	0x0_0007

After clearing CCIF to launch the Verify Backdoor Access Key command, the flash memory module checks the FSEC[KEYEN] bits to verify that this command is enabled. If not enabled, the flash memory module sets the FSTAT[ACCERR] bit and terminates. If the command is enabled, the flash memory module compares the key provided in FCCOB to the backdoor comparison key in the Flash Configuration Field. If the backdoor keys match, the FSEC[SEC] field is changed to the unsecure state and security is 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 immediately aborted and the FSTAT[ACCERR] bit is (again) set to 1 until a reset of the flash memory module module occurs. If the entire 8-byte key is all zeros or all ones, the Verify Backdoor Access Key command fails with an access error. The CCIF flag is set after the Verify Backdoor Access Key operation completes.

Table 20-40. Verify Backdoor Access Key Command Error Handling

Error Condition	Error Bit
The supplied key is all-0s or all-Fs	FSTAT[ACCERR]
An incorrect backdoor key is supplied	FSTAT[ACCERR]
Backdoor key access has not been enabled (see the description of the FSEC register)	FSTAT[ACCERR]
This command is launched and the backdoor key has mismatched since the last power down reset	FSTAT[ACCERR]

20.4.11.14 Program Partition Command

The Program Partition command prepares the FlexNVM block for use as data flash, EEPROM backup, or a combination of both and initializes the FlexRAM. The Program Partition command must not be launched from flash memory, since flash memory resources are not accessible during Program Partition command execution.

CAUTION

While different partitions of the FlexNVM are available, the intention is that a single partition choice is used throughout the entire lifetime of a given application. The FlexNVM Partition Code choices affect the endurance and data retention characteristics of the device.

Table 20-41. Program Partition Command FCCOB Requirements

FCCOB Number	FCCOB Contents [7:0]
0	0x80 (PGMPART)
1	Not Used

Table continues on the next page...

Table 20-41. Program Partition Command FCCOB Requirements (continued)

FCCOB Number	FCCOB Contents [7:0]
2	Not Used
3	Not Used
4	EEPROM Data Set Size Code[1]
5	FlexNVM Partition Code[2]

1. See Table 2 and EEPROM Data Set Size
1. See Table 2 and EEPROM Data Set Size
1. See Table 2 and EEPROM Data Set Size
2. See Table 3
2. See Table 3

Table 20-42. Valid EEPROM Data Set Size Codes

EEPROM Data Set Size Code (FCCOB4)[1]		EEPROM Data Set Size (Bytes)
FCCOB4[5:4]	FCCOB4[EEESIZE]	
11	0xF	0[2]
11	0x9	32
11	0x8	64
11	0x7	128
11	0x6	256
11	0x5	512
11	0x4	1024
11	0x3	2048

1. FCCOB4[7:6] = 00
2. EEPROM Data Set Size must be set to 0 bytes when the FlexNVM Partition Code is set for no EEPROM.

Table 20-43. Valid FlexNVM Partition Codes

FlexNVM Partition Code (FCCOB5[DEPART])[1]	Data flash Size (KB)	EEPROM backup Size (KB)
0000	32	0
0001	24	8
0010	16	16
0011	0	32
1000	0	32
1001	8	24
1010	16	16
1011	32	0

1. FCCOB5[7:4] = 0000

After clearing CCIF to launch the Program Partition command, the flash memory module first verifies that the EEPROM Data Set Size Code and FlexNVM Partition Code in the data flash IFR are erased. If erased, the Program Partition command erases the contents

of the FlexNVM memory. If the FlexNVM is to be partitioned for EEPROM backup, the allocated EEPROM backup sectors are formatted for EEPROM use. Finally, the partition codes are programmed into the data flash IFR using the values provided. The Program Partition command also verifies that the partition codes read back correctly after programming. If the FlexNVM is partitioned for EEPROM backup, the EEERDY flag will set with RAMRDY clear. If the FlexNVM is not partitioned for EEPROM backup, the RAMRDY flag will set with EEERDY clear. The CCIF flag is set after the Program Partition operation completes.

Prior to launching the Program Partition command, the data flash IFR must be in an erased state, which can be accomplished by executing the Erase All Blocks command or by an external request (see Erase All Blocks Command). The EEPROM Data Set Size Code and FlexNVM Partition Code are read using the Read Resource command (see Read Resource Command).

Table 20-44. Program Partition Command Error Handling

Error Condition	Error Bit
Command not available in current mode/security	FSTAT[ACCERR]
The EEPROM data size and FlexNVM partition code bytes are not initially 0xFFFF	FSTAT[ACCERR]
Invalid EEPROM Data Set Size Code is entered (see Table 20-42 for valid codes)	FSTAT[ACCERR]
Invalid FlexNVM Partition Code is entered (see Table 20-43 for valid codes)	FSTAT[ACCERR]
FlexNVM Partition Code = full data flash (no EEPROM) and EEPROM Data Set Size Code allocates FlexRAM for EEPROM	FSTAT[ACCERR]
FlexNVM Partition Code allocates space for EEPROM backup, but EEPROM Data Set Size Code allocates no FlexRAM for EEPROM	FSTAT[ACCERR]
FCCOB4[7:6] != 00	FSTAT[ACCERR]
FCCOB5[7:4] != 0000	FSTAT[ACCERR]
Any errors have been encountered during the verify operation	FSTAT[MGSTAT0]

20.4.11.15 Set FlexRAM Function Command

The Set FlexRAM Function command changes the function of the FlexRAM:

- When not partitioned for EEPROM, the FlexRAM is typically used as traditional RAM.
- When partitioned for EEPROM, the FlexRAM is typically used to store EEPROM data.

Table 20-45. Set FlexRAM Function Command FCCOB Requirements

FCCOB Number	FCCOB Contents [7:0]
0	0x81 (SETRAM)
1	FlexRAM Function Control Code

Table 20-45. Set FlexRAM Function Command FCCOB Requirements

FCCOB Number	FCCOB Contents [7:0]
	(see Table 20-46)

Table 20-46. FlexRAM Function Control

FlexRAM Function Control Code	Action
0xFF	Make FlexRAM available as RAM: • Clear the FCNFG[EEERDY] and FCNFG[RAMRDY] flags • Write a background of ones to all FlexRAM locations • Set the FCNFG[RAMRDY] flag
0x00	Make FlexRAM available for EEPROM: • Clear the FCNFG[EEERDY] and FCNFG[RAMRDY] flags • Write a background of ones to all FlexRAM locations • Copy-down existing EEPROM data to FlexRAM • Set the FCNFG[EEERDY] flag

After clearing CCIF to launch the Set FlexRAM Function command, the flash memory module sets the function of the FlexRAM based on the FlexRAM Function Control Code.

When making the FlexRAM available as traditional RAM, the flash memory module clears the FCNFG[EEERDY] and FCNFG[RAMRDY] flags, overwrites the contents of the entire FlexRAM with a background pattern of all ones, and sets the FCNFG[RAMRDY] flag. The state of the FEPROT register does not prevent the FlexRAM from being overwritten. When the FlexRAM is set to function as a RAM, normal read and write accesses to the FlexRAM are available. When large sections of flash memory need to be programmed, e.g. during factory programming, the FlexRAM can be used as the Section Program Buffer for the Program Section command (see Program Section Command).

When making the FlexRAM available for EEPROM, the flash memory module clears the FCNFG[EEERDY] and FCNFG[RAMRDY] flags, overwrites the contents of the FlexRAM allocated for EEPROM with a background pattern of all ones, and copies the existing EEPROM data from the EEPROM backup record space to the FlexRAM. After completion of the EEPROM copy-down, the FCNFG[EEERDY] flag is set. When the FlexRAM is set to function as EEPROM, normal read and write access to the FlexRAM is available, but writes to the FlexRAM also invoke EEPROM activity. The CCIF flag is set after the Set FlexRAM Function operation completes.

Table 20-47. Set FlexRAM Function Command Error Handling

Error Condition	Error Bit
Command not available in current mode/security	FSTAT[ACCERR]

Table continues on the next page...

Table 20-47. Set FlexRAM Function Command Error Handling (continued)

Error Condition	Error Bit
FlexRAM Function Control Code is not defined	FSTAT[ACCERR]
FlexRAM Function Control Code is set to make the FlexRAM available for EEPROM, but FlexNVM is not partitioned for EEPROM	FSTAT[ACCERR]

20.4.12 Security

The flash memory module provides security information to the MCU based on contents of the FSEC security register.

The MCU then limits access to flash memory resources as defined in the device's Chip Configuration details. During reset, the flash memory module initializes the FSEC register using data read from the security byte of the Flash Configuration Field (see Flash Configuration Field Description).

The following fields are available in the FSEC register. The settings are described in the Flash Security Register (FTFL_FSEC) details.

Flash security features are discussed further in AN4507: Using the Kinetis Security and Flash Protection Features . Note that not all features described in the application note are available on this device.

Table 20-48. FSEC register fields

FSEC field	Description
KEYEN	Backdoor Key Access
MEEN	Mass Erase Capability
FSLACC	Factory Security Level Access
SEC	MCU security

20.4.12.1 Flash Memory Access by Mode and Security

The following table summarizes how access to the flash memory module is affected by security and operating mode.

Table 20-49. Flash Memory Access Summary

Operating Mode	Chip Security State	
	Unsecure	Secure
NVM Normal	Full command set	
NVM Special	Full command set	Only the Erase All Blocks and Read 1s All Blocks commands.

20.4.12.2 Changing the Security State

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 program flash erase and program commands are available and that the region of the program flash containing the flash configuration field is unprotected. If the flash security byte is successfully programmed, its new value takes affect after the next chip reset.

20.4.12.2.1 Unsecuring the Chip Using Backdoor Key Access

The chip can be unsecured by using the backdoor key access feature, which requires knowledge of the contents of the 8-byte backdoor key value stored in the Flash Configuration Field (see Flash Configuration Field Description). If the FSEC[KEYEN] bits are in the enabled state, the Verify Backdoor Access Key command (see Verify Backdoor Access Key Command) can be run; it allows the user to present prospective keys for comparison to the stored keys. If the keys match, the FSEC[SEC] bits are changed to unsecure the chip. The entire 8-byte key cannot be all 0s or all 1s; that is, 0000_0000_0000_0000h and FFFF_FFFF_FFFF_FFFFh are not accepted by the Verify Backdoor Access Key command as valid comparison values. While the Verify Backdoor Access Key command is active, program flash memory is not available for read access and returns invalid data.

The user code stored in the program 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 bits are in the enabled state, the chip can be unsecured by the following backdoor key access sequence:

1. Follow the command sequence for the Verify Backdoor Access Key command as explained in Verify Backdoor Access Key Command

2. If the Verify Backdoor Access Key command is successful, the chip is unsecured and the FSEC[SEC] bits are forced to the unsecure state

An illegal key provided to the Verify Backdoor Access Key command prohibits further use of the Verify Backdoor Access Key command. A reset of the chip is the only method to re-enable the Verify Backdoor Access Key command when a comparison fails.

After the backdoor keys have been correctly matched, the chip is unsecured by changing the FSEC[SEC] bits. A successful execution of the Verify Backdoor Access Key command changes the security in the FSEC register only. It does not alter the security byte or the keys stored in the Flash Configuration Field (Flash Configuration Field Description). After the next reset of the chip, the security state of the flash memory module reverts back to the flash security byte in the Flash Configuration Field. The Verify Backdoor Access Key command sequence has no effect on the program and erase protections defined in the program flash protection registers.

If the backdoor keys successfully match, the unsecured chip has full control of the contents of the Flash Configuration Field. The chip may erase the sector containing the Flash Configuration Field and reprogram the flash security byte to the unsecure state and change the backdoor keys to any desired value.

20.4.13 Reset Sequence

On each system reset the flash memory module executes a sequence which establishes initial values for the flash block configuration parameters, FPROT, FDPROT, FEPROT, FOPT, and FSEC registers and the FCNFG[RAMRDY, EEERDY] bits.

FSTAT[CCIF] is cleared throughout the reset sequence. The flash memory module holds off CPU access during the reset sequence. Flash reads are possible when the hold is removed. Completion of the reset sequence is marked by setting CCIF which enables flash user commands.

If a reset occurs while any flash command is in progress, that command is immediately aborted. The state of the word being programmed or the sector/block being erased is not guaranteed. Commands and operations do not automatically resume after exiting reset.

Chapter 21
Computer Operating Properly (COP) Watchdog

21.1 Introduction

The computer operating properly (COP) module is used to help software recover from runaway code. The COP is a free-running down counter that, once enabled, is designed to generate a reset upon reaching zero. Software must periodically service the COP in order to reload the counter and prevent a reset.

21.1.1 Features

The COP module includes these distinctive features:

- Programmable prescaler
- Programmable timeout period = (cop_prescaler*(TIMEOUT + 1)) clock cycles, where TIMEOUT can be from 0x0000 to 0xFFFF
- Programmable interrupt timing that can occur for any count less than the TIMEOUT value
- Programmable wait and stop operation
- COP timer is disabled while the core is in debug mode
- Causes loss of reference reset 128 cycles after loss of reference clock to the PLL is detected
- Choice of clock sources for counter
- Integrated low speed oscillator

21.1.2 Block Diagram

The block diagram of the COP module follows.

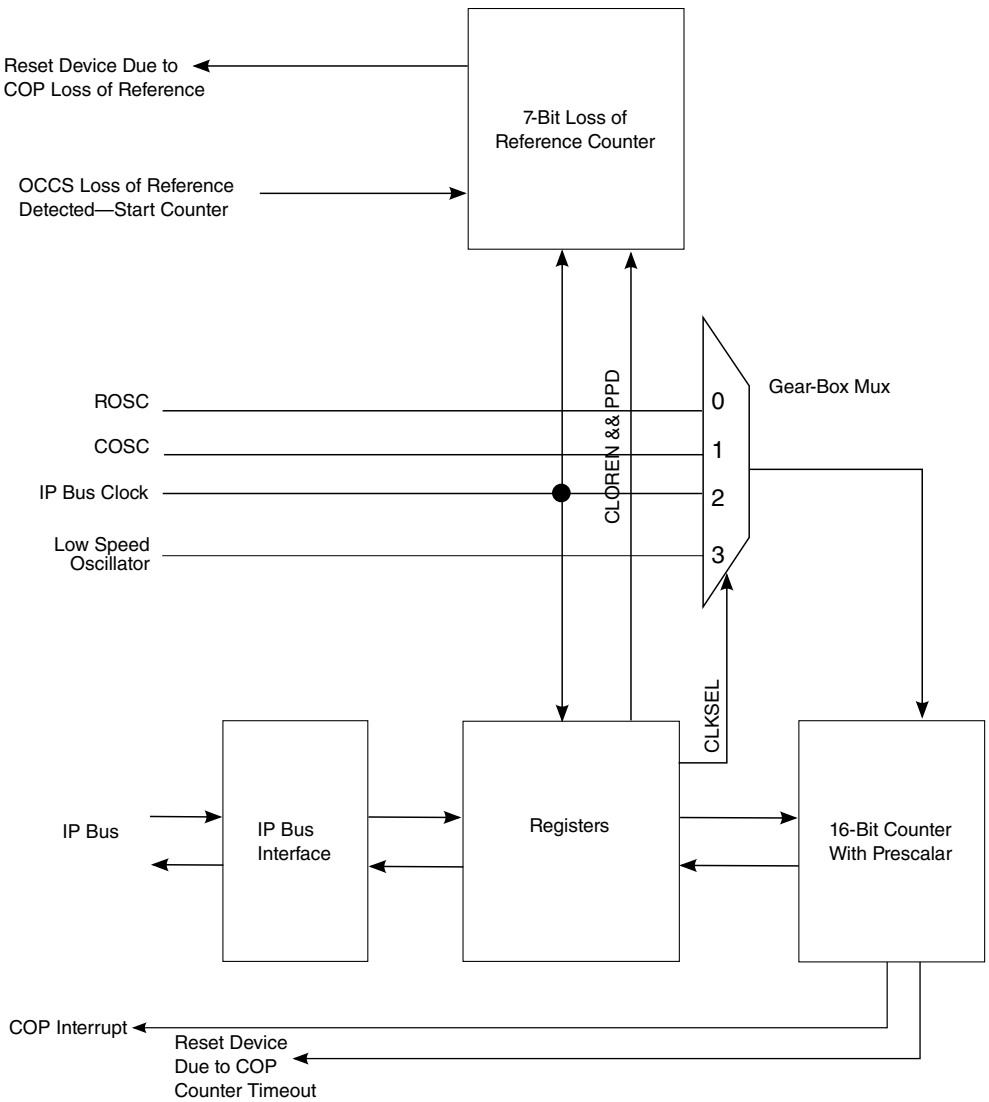


Figure 21-1. COP Module Block Diagram with Low Speed Clock

21.2 Memory Map and Registers

COP memory map

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
E320	COP Control Register (COP_CTRL)	16	R/W	0302h	21.2.1/439

Table continues on the next page...

COP memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
E321	COP Timeout Register (COP_TOUT)	16	R/W	FFFFh	21.2.2/441
E322	COP Counter Register (COP_CNTR)	16	R/W	FFFFh	21.2.3/441
E323	COP Interrupt Value Register (COP_INTVAL)	16	R/W	00FFh	21.2.4/442

21.2.1 COP Control Register (COP_CTRL)

Address: E320h base + 0h offset = E320h

Bit	15	14	13	12	11	10	9	8
Read	0						PSS	
Write								
Reset	0	0	0	0	0	0	1	1

Bit	7	6	5	4	3	2	1	0
Read Write	INTEN	CLKSEL		CLOREN	CSEN	CWEN	CEN	CWP
Reset	0	0	0	0	0	0	1	0

COP_CTRL field descriptions

Field	Description
15–10 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
9–8 PSS	Prescaler Select This two bit field determines the value of the clock divider (prescaler). You may divide the source clock by 1, 16, 256, or 1024. Generally, you use a lower prescaler value for lower frequency clock sources, but any combination of PSS and timeout may be used as long as they yield the desired timeout value. **Restriction:** This field can be changed only when CWP is set to zero. 00 No division 01 Divide by 16 10 Divide by 256 11 Divide by 1024
7 INTEN	Interrupt Enable This bit is used to enable an interrupt when the counter value reaches the value of the INTVAL register. Clear the interrupt by servicing the counter, disabling the COP, or clearing this bit. **Restriction:** This field can be changed only when CWP is set to zero. 0 COP interrupt is disabled. (default) 1 COP interrupt is enabled.
6–5 CLKSEL	Clock Source Select This bitfield selects the clock source for the COP counter. Some safety applications require the watchdog counter to use a clock source different than the system clock.

Table continues on the next page...

COP_CTRL field descriptions (continued)

Field	Description
	Restriction: This field can be changed only when CWP is set to zero. It also should be changed only when CEN is clear. 00 Relaxation oscillator output (ROSC) is used to clock the counter (default) 01 Crystal oscillator output (COSC) is used to clock the counter 10 IP bus clock is used to clock the counter **Restriction:** Do not select the IP bus clock to clock the counter if the application requires the COP to wake the device from stop mode. 11 Low speed oscillator is used to clock the counter
4 CLOREN	COP Loss of Reference Enable This bit enables the operation of the COP loss of reference counter. **Restriction:** This bit can be changed only when CWP is set to zero. 0 COP loss of reference counter is disabled. (default) 1 COP loss of reference counter is enabled.
3 CSEN	COP Stop Mode Enable This bit controls the operation of the COP counter in stop mode. **Restriction:** This bit can be changed only when CWP is set to zero. 0 COP counter stops in stop mode. (default) 1 COP counter runs in stop mode if CEN is set to one.
2 CWEN	COP Wait Mode Enable This bit controls the operation of the COP counter in wait mode. **Restriction:** This bit can be changed only when CWP is set to zero. 0 COP counter stops in wait mode. (default) 1 COP counter runs in wait mode if CEN is set to one.
1 CEN	COP Enable This bit controls the operation of the COP counter. This bit always reads as zero when the chip is in debug mode. **Restriction:** This bit can be changed only when CWP is set to zero. 0 COP counter is disabled. 1 COP counter is enabled. (default)
0 CWP	COP Write Protect This bit controls the write protection feature of the COP control (CTRL) register, the COP interrupt value (INTVAL) register and the COP timeout (TOUT) register. Once set, this bit can be cleared only by resetting the module. 0 The CTRL, INTVAL and TOUT registers are readable and writable. (default) 1 The CTRL, INTVAL and TOUT registers are read-only.

21.2.2 COP Timeout Register (COP_TOUT)

The value in this register determines the timeout period of the COP counter.

Considerations about setting the timeout value follow:

1. TIMEOUT should be written before the COP is enabled. Changing TIMEOUT while the COP is enabled results in a timeout period that differs from the expected value.
2. After the COP has been enabled:
   - The recommended procedure for changing TIMEOUT is to disable the COP, write to TOUT, and then re-enable the COP. This procedure ensures that the new TIMEOUT is loaded into the counter.
   - Alternatively, the CPU can write to TOUT and then write the proper patterns to CNTR to cause the counter to reload with the new TIMEOUT value.
3. A minimum time difference is required between the TIMEOUT value and the value of INTVAL[INTERRUPT_VALUE]. When the bus clock is the source for the COP counter, a typical minimum time difference is 40 cycles.

Address: E320h base + 1h offset = E321h

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read Write	TIMEOUT															
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

COP_TOUT field descriptions

Field	Description
TIMEOUT	COP Timeout Period The value in this register determines the timeout period of the COP counter. **Restriction:** These bits can be changed only when CWP is set to zero.

21.2.3 COP Counter Register (COP_CNTR)

Address: E320h base + 2h offset = E322h

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read Write	COUNT_SERVICE															
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

COP_CNTR field descriptions

Field	Description
COUNT_ SERVICE	COP Count/Service COP Count: When this register is read, its value is the current value of the COP counter as it counts down from the timeout value to zero. A reset is issued when this count reaches zero.

COP_CNTR field descriptions (continued)

Field	Description
	COP Service: Write to this register to service the counter. When enabled, the COP requires that a service sequence be performed periodically to clear the COP counter and prevent a reset from being issued. • This routine consists of writing 0x5555 to CNTR followed by writing 0xAAAA before the timeout period expires. • These writes to CNTR must be performed in the correct order, but any number of other instructions (and writes to other registers) may be executed between the two writes.

21.2.4 COP Interrupt Value Register (COP_INTVAL)

Address: E320h base + 3h offset = E323h

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read Write	INTERRUPT_VALUE															
Reset	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1

COP_INTVAL field descriptions

Field	Description
INTERRUPT_ VALUE	COP Interrupt Value When the count value is equal to this interrupt value, an interrupt is generated if it is enabled via the CTRL[INTEN] bit. The interrupt can be cleared by perfoming the service routine to the counter, by disabling the COP (setting the CTRL[CEN] bit to 0), or by clearing the interrupt enable (setting the CTRL[INTEN] bit to 0). Servicing the counter requires time to resynchronize to the clock used by the counter. Allow time for this resynchronization to occur before exiting the interrupt service routine; otherwise, a new interrupt may be issued for the same event. To confirm that the resychronization has occurred, verify that CNTR is greater than INTVAL prior to exiting the ISR. **Restriction:** Do not change this field's value while the counter is enabled. **Restriction:** These bits can be changed only when the CTRL[CWP] bit is 0.

21.3 Functional Description

When the COP is enabled, each positive edge of the prescaled clock (COSC, ROSC, low speed oscillator, or IP Bus clock) causes the counter to decrement by one. If the count reaches a value equal to the INTVAL register, then an interrupt may be issued. If the count reaches a value of 0x0000, then the device is reset. For the core to show that it is operating properly, it must perform a service routine before the count reaches 0x0000. The service routine consists of writing 0x5555 followed by 0xAAAA to the CNTR register. This service routine also clears the interrupt.

21.3.1 COP after Reset

CEN is set out of reset. Thus the counter is enabled by default. In addition, the TOUT register is set to its maximum value of 0xFFFF and PRESCALER is set to 1024 (CTRL[PSS]=11) during reset so the counter is loaded with a maximum timeout period when reset is released.

If the IP bus clock to the COP is not enabled by default after reset, then allow 2 clock cycles to occur after enabling it before performing a write access to the COP.

21.3.2 Wait Mode Operation

If wait mode is entered with both CEN and CWEN set to 1, then the COP counter continues to count down. In that case, a COP reset is issued to wake the device after the counter reaches zero. A COP interrupt may wake the device prior to the counter reaching zero if the interrupt is properly programmed and enabled. If either CEN or CWEN is cleared to 0 when wait mode is entered, then the counter is disabled and reloads using the value in the TOUT register.

21.3.3 Stop Mode Operation

If stop mode is entered with both CEN and CSEN set to 1, then the COP counter continues to count down. In that case, a COP reset is issued to wake the device after the counter reaches zero. A COP interrupt may wake the device prior to the counter reaching zero if the interrupt is properly programmed and enabled. If either CEN or CSEN is cleared to 0 when stop mode is entered, then the counter is disabled and reloads using the value in the TOUT register.

21.3.4 Debug Mode Operation

The COP counter is not allowed to count when the device is in debug mode. In addition, the CEN bit in the CTRL register always reads as zero when the device is in debug. The actual value of CEN is unaffected by debug, however, and resumes its previously set value upon exiting debug.

21.3.5 Loss of Reference Operation

When the OCCS signals the COP that a loss of the reference clock has occurred and the CLOREN bit is set, then the COP starts a 7-bit counter that runs off of the IP bus clock (which continues to be produced by the PLL for at least 1000 cycles upon losing its reference). The counter continues to count even if the loss of reference clock condition goes away. When this counter reaches 0x7F, it causes a loss of reference reset that resets the entire device. If the software has safely shut down the device and does not want a full reset, then the loss of reference timeout count can be delayed by servicing the COP counter in the standard manner of writing 0x5555 followed by 0xAAAA or stopped by setting CLOREN to zero.

21.4 Resets

Any system reset forces all registers to their reset state and clears the COP_RST_B or LOR_RST_B signals if they are asserted. The counter is loaded with its maximum value of 0xFFFF, the prescaler is set to 1024, and the counter restarts when reset is released because CEN is enabled by default.

21.5 Clocks

The COP timer base is the COSC, ROSC, low speed oscillator, or IP Bus clock divided by the prescaler value (1 - 1024).

21.6 Interrupts

The COP module generates a single interrupt that occurs when the counter matches the value of the INTVAL register. Enabling this interrupt is controlled by CTRL[INTEN].

The COP module generates the chip-wide COP reset signal when the counter reaches a value of 0x0000. It can also generate a chip-wide reset signal 128 IP Bus cycles after the loss of the reference clock is detected.

NOTE

The chip reset vector base address is located at P:00h. The COP reset vector base address is located at P:02h. For more details, refer to the interrupt vector table.

Chapter 22
External Watchdog Monitor (EWM)

22.1 Introduction

NOTE

For the chip-specific implementation details of this module's instances, see the chip configuration information.

The watchdog is generally used to monitor the flow and execution of embedded software within an MCU. The watchdog consists of a counter that if allowed to overflow, forces an internal reset (asynchronous) to all on-chip peripherals and optionally assert the $\overline{\text{RESET}}$ pin to reset external devices/circuits. The overflow of the watchdog counter must not occur if the software code works well and services the watchdog to re-start the actual counter.

For safety, a redundant watchdog system, External Watchdog Monitor (EWM), is designed to monitor external circuits, as well as the MCU software flow. This provides a back-up mechanism to the internal watchdog that resets the MCU's CPU and peripherals.

The EWM differs from the internal watchdog in that it does not reset the MCU's CPU and peripherals. The EWM if allowed to time-out, provides an independent $\overline{\text{EWM_out}}$ pin that when asserted resets or places an external circuit into a safe mode. The CPU resets the EWM counter that is logically ANDed with an external digital input pin. This pin allows an external circuit to influence the reset_out signal.

22.1.1 Features

Features of EWM module include:

- Independent LPO clock source
- Programmable time-out period specified in terms of number of EWM LPO clock cycles.

- Windowed refresh option
  - Provides robust check that program flow is faster than expected.
  - Programmable window.
  - Refresh outside window leads to assertion of $\overline{\text{EWM_out}}$.
- Robust refresh mechanism
  - Write values of 0xB4 and 0x2C to EWM Refresh Register within 15 (*EWM_service_time*) peripheral bus clock cycles.
- One output port, $\overline{\text{EWM_out}}$, when asserted is used to reset or place the external circuit into safe mode.
- One Input port, EWM_in, allows an external circuit to control the $\overline{\text{EWM_out}}$ signal.

22.1.2 Modes of Operation

This section describes the module's operating modes.

22.1.2.1 Stop Mode

When the EWM is in stop mode, the CPU services to the EWM cannot occur. On entry to stop mode, the EWM's counter freezes.

There are two possible ways to exit from Stop mode:

- On exit from stop mode through a reset, the EWM remains disabled.
- On exit from stop mode by an interrupt, the EWM is re-enabled, and the counter continues to be clocked from the same value prior to entry to stop mode.

Note the following if the EWM enters the stop mode during CPU service mechanism: At the exit from stop mode by an interrupt, refresh mechanism state machine starts from the previous state which means, if first service command is written correctly and EWM enters the stop mode immediately, the next command has to be written within the next 15 (*EWM_service_time*) peripheral bus clocks after exiting from stop mode. User must mask all interrupts prior to executing EWM service instructions.

22.1.2.2 Wait Mode

The EWM module treats the stop and wait modes as the same. EWM functionality remains the same in both of these modes.

22.1.2.3 Debug Mode

Entry to debug mode has no effect on the EWM.

- If the EWM is enabled prior to entry of debug mode, it remains enabled.
- If the EWM is disabled prior to entry of debug mode, it remains disabled.

22.1.3 Block Diagram

This figure shows the EWM block diagram.

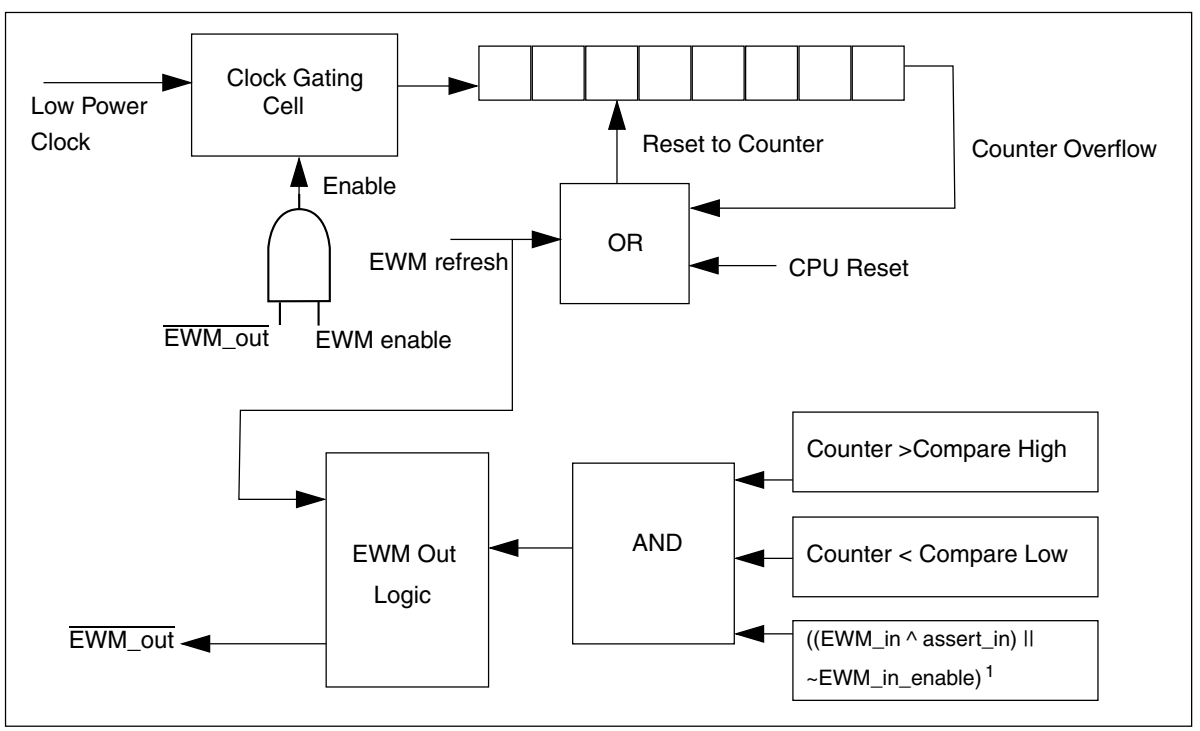


Figure 22-1. EWM Block Diagram

22.2 EWM Signal Descriptions

The EWM has two external signals and internal options for the counter clock sources, as shown in the following table.

Table 22-1. EWM Signal Descriptions

Signal	Description	I/O
EWM_in	EWM input for safety status of external safety circuits. The polarity of EWM_in is programmable using the EWM_CTRL[ASSIN] bit. The default polarity is active-low.	I
$\overline{\text{EWM_out}}$	EWM reset out signal	O
lpo_clk[3:0]	Low power clock sources for running counter	I

22.3 Memory Map/Register Definition

This section contains the module memory map and registers.

NOTE

Each 8-bit register occupies bits 0-7 of a 16-bit width. The other 8 bits are read-only and always read 0.

EWM memory map

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
E330	Control Register (EWM_CTRL)	16	R/W	0000h	22.3.1/448
E331	Service Register (EWM_SERV)	16	W (always reads 0)	0000h	22.3.2/450
E332	Compare Low Register (EWM_CMPL)	16	R/W	0000h	22.3.3/450
E333	Compare High Register (EWM_CMPH)	16	R/W	00FFh	22.3.4/451
E334	Clock Control Register (EWM_CLKCTRL)	16	R/W	0000h	22.3.5/451
E335	Clock Prescaler Register (EWM_CLKPRESCALER)	16	R/W	0000h	22.3.6/452

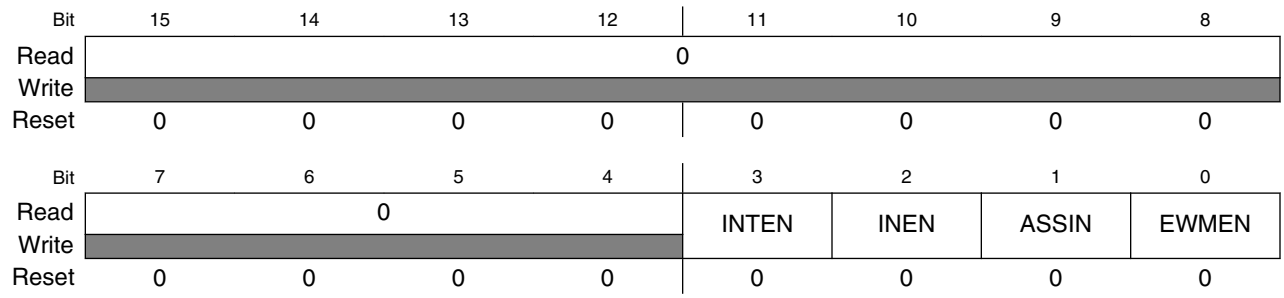
22.3.1 Control Register (EWM_CTRL)

The CTRL register is cleared by any reset.

NOTE

INEN, ASSIN and EWMEN bits can be written once after a CPU reset. Modifying these bits more than once, generates a bus transfer error.

Address: E330h base + 0h offset = E330h



EWM_CTRL field descriptions

Field	Description
15–8 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
7–4 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
3 INTEN	Interrupt Enable. This bit when set and $\overline{\text{EWM_out}}$ is asserted, an interrupt request is generated. To de-assert interrupt request, user should clear this bit by writing 0.
2 INEN	Input Enable. This bit when set, enables the EWM_in port.
1 ASSIN	EWM_in's Assertion State Select. Default assert state of the EWM_in signal is logic zero. Setting ASSIN bit inverts the assert state to a logic one.
0 EWMEN	EWM enable. This bit when set, enables the EWM module. This resets the EWM counter to zero and deasserts the $\overline{\text{EWM_out}}$ signal. Clearing EWMEN bit disables the EWM, and therefore it cannot be enabled until a reset occurs, due to the write-once nature of this bit.

22.3.2 Service Register (EWM_SERV)

The SERV register provides the interface from the CPU to the EWM module. It is write-only and reads of this register return zero.

Address: E330h base + 1h offset = E331h

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	0								0							
Write									SERVICE							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

EWM_SERV field descriptions

Field	Description
15–8 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
SERVICE	The EWM service mechanism requires the CPU to write two values to the SERV register: a first data byte of 0xB4, followed by a second data byte of 0x2C. The EWM service is illegal if either of the following conditions is true. • The first or second data byte is not written correctly. • The second data byte is not written within a fixed number of peripheral bus cycles of the first data byte. This fixed number of cycles is called *EWM_service_time*.

22.3.3 Compare Low Register (EWM_CMPL)

The CMPL register is reset to zero after a CPU reset. This provides no minimum time for the CPU to service the EWM counter.

NOTE

This register can be written only once after a CPU reset. Writing this register more than once generates a bus transfer error.

Address: E330h base + 2h offset = E332h

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	0								COMPAREL							
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

EWM_CMPL field descriptions

Field	Description
15–8 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

Table continues on the next page...

EWM_CMPL field descriptions (continued)

Field	Description
COMPAREL	To prevent runaway code from changing this field, software should write to this field after a CPU reset even if the (default) minimum service time is required.

22.3.4 Compare High Register (EWM_CMPH)

The CMPH register is reset to 0x00FF after a CPU reset. This provides a maximum of 256 clocks time, for the CPU to service the EWM counter.

NOTE

This register can be written only once after a CPU reset. Writing this register more than once generates a bus transfer error.

NOTE

The valid values for CMPH are up to 0x00FE because the EWM counter never expires when CMPH = 0x00FF. The expiration happens only if EWM counter is greater than CMPH.

Address: E330h base + 3h offset = E333h

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	0								COMPAREH							
Write																
Reset	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1

EWM_CMPH field descriptions

Field	Description
15–8 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
COMPAREH	To prevent runaway code from changing this field, software should write to this field after a CPU reset even if the (default) maximum service time is required.

22.3.5 Clock Control Register (EWM_CLKCTRL)

This CLKCTRL register is reset to 0x00 after a CPU reset.

NOTE

This register can be written only once after a CPU reset. Writing this register more than once generates a bus transfer error.

NOTE

User should select the required low power clock before enabling the EWM.

Address: E330h base + 4h offset = E334h

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	0								0						CLKSEL	
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

EWM_CLKCTRL field descriptions

Field	Description
15–8 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
7–2 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
CLKSEL	EWM has 4 possible low power clock sources for running EWM counter. One of the clock source can be selected by writing into this field. • 00 - lpo_clk[0] will be selected for running EWM counter. • 01 - lpo_clk[1] will be selected for running EWM counter. • 10 - lpo_clk[2] will be selected for running EWM counter. • 11 - lpo_clk[3] will be selected for running EWM counter.

22.3.6 Clock Prescaler Register (EWM_CLKPRESCALER)

This CLKPRESCALER register is reset to 0x00 after a CPU reset.

NOTE

This register can be written only once after a CPU reset. Writing this register more than once generates a bus transfer error.

NOTE

Write the required prescaler value before enabling the EWM.

NOTE

The implementation of this register is chip-specific. See the Chip Configuration details.

Address: E330h base + 5h offset = E335h

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	0								CLK_DIV							
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

EWM_CLKPRESCALER field descriptions

Field	Description
15–8 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
CLK_DIV	Selected low power source for running the EWM counter can be prescaled as below. • Prescaled clock frequency = low power clock source frequency/ (1+ CLK_DIV)

22.4 Functional Description

The following sections describe functional details of the EWM module.

22.4.1 The $\overline{\text{EWM_out}}$ Signal

The $\overline{\text{EWM_out}}$ is a digital output signal used to gate an external circuit (application specific) that controls critical safety functions.

The $\overline{\text{EWM_out}}$ signal remains deasserted when the EWM is being regularly serviced by the CPU within the programmable service window, indicating that the application code is executed as expected.

The $\overline{\text{EWM_out}}$ signal is asserted in any of the following conditions:

- Servicing the EWM when the counter value is less than CMPL value.
- If the EWM counter value reaches the CMPH value, and no EWM service has occurred.
- Servicing the EWM when the counter value is more than CMPL and less than CMPH values and EWM_in signal is asserted.
- If functionality of EWM_in pin is enabled and EWM_in pin is asserted while servicing the EWM.
- After any reset (by the virtue of the external pull-down mechanism on the $\overline{\text{EWM_out}}$ pin)

On a normal reset, the $\overline{\text{EWM_out}}$ is asserted. To deassert the $\overline{\text{EWM_out}}$, set EWMEN bit in the CTRL register to enable the EWM.

If the $\overline{\text{EWM_out}}$ signal shares its pad with a digital I/O pin, on reset this actual pad defers to being an input signal. It takes the $\overline{\text{EWM_out}}$ output condition only after you enable the EWM by the EWMEN bit in the CTRL register.

When the EWM_out pin is asserted, it can only be deasserted by forcing a MCU reset.

Note

EWM_out pad must be in pull down state when EWM functionality is used and when EWM is under Reset.

22.4.2 The EWM_in Signal

The EWM_in is a digital input signal that allows an external circuit to control the EWM_out signal. For example, in the application, an external circuit monitors a critical safety function, and if there is fault with this circuit's behavior, it can then actively initiate the EWM_out signal that controls the gating circuit.

The EWM_in signal is ignored if the EWM is disabled, or if INEN bit of CTRL register is cleared, as after any reset.

On enabling the EWM (setting the CTRL[EWMEN] bit) and enabling EWM_in functionality (setting the CTRL[INEN] bit), the EWM_in signal must be in the deasserted state prior to the CPU servicing the EWM. This ensures that the EWM_out stays in the deasserted state; otherwise, the EWM_out pin is asserted.

Note

You must update the CMPH and CMPL registers prior to enabling the EWM. After enabling the EWM, the counter resets to zero, therefore providing a reasonable time after a power-on reset for the external monitoring circuit to stabilize and ensure that the EWM_in pin is deasserted.

22.4.3 EWM Counter

It is an 8-bit ripple counter fed from a clock source that is independent of the peripheral bus clock source. As the preferred time-out is between 1 ms and 100 ms the actual clock source should be in the kHz range.

The counter is reset to zero, after a CPU reset, or a EWM refresh cycle. The counter value is not accessible to the CPU.

22.4.4 EWM Compare Registers

The compare registers CMPL and CMPH are write-once after a CPU reset and cannot be modified until another CPU reset occurs.

The EWM compare registers are used to create a service window, which is used by the CPU to service/refresh the EWM module.

- If the CPU services the EWM when the counter value lies between CMPL value and CMPH value, the counter is reset to zero. This is a legal service operation.
- If the CPU executes a EWM service/refresh action outside the legal service window, $\overline{\text{EWM_out}}$ is asserted.

It is illegal to program CMPL and CMPH with same value. In this case, as soon as counter reaches (CMPL + 1), $\overline{\text{EWM_out}}$ is asserted.

22.4.5 EWM Refresh Mechanism

Other than the initial configuration of the EWM, the CPU can only access the EWM by the EWM Service Register. The CPU must access the EWM service register with correct write of unique data within the windowed time frame as determined by the CMPL and CMPH registers. Therefore, three possible conditions can occur:

Table 22-2. EWM Refresh Mechanisms

Condition	Mechanism
A unique EWM service occurs when CMPL < Counter < CMPH.	The software behaves as expected and the counter of the EWM is reset to zero, and $\overline{\text{EWM_out}}$ pin remains in the deasserted state. **Note:** EWM_in pin is also assumed to be in the deasserted state.
A unique EWM service occurs when Counter < CMPL	The software services the EWM and therefore resets the counter to zero and asserts the $\overline{\text{EWM_out}}$ pin (irrespective of the EWM_in pin). The $\overline{\text{EWM_out}}$ pin is expected to gate critical safety circuits.
Counter value reaches CMPH prior to a unique EWM service	The counter value reaches the CMPH value and no service of the EWM resets the counter to zero and assert the $\overline{\text{EWM_out}}$ pin (irrespective of the EWM_in pin). The $\overline{\text{EWM_out}}$ pin is expected to gate critical safety circuits.

Any illegal service on EWM has no effect on $\overline{\text{EWM_out}}$.

22.4.6 EWM Interrupt

When $\overline{\text{EWM_out}}$ is asserted, an interrupt request is generated to indicate the assertion of the EWM reset out signal. This interrupt is enabled when CTRL[INTEN] is set. Clearing this bit clears the interrupt request but does not affect $\overline{\text{EWM_out}}$. The $\overline{\text{EWM_out}}$ signal can be deasserted only by forcing a system reset.

22.4.7 Selecting the EWM counter clock

There are four possible low power clock sources for the EWM counter. Select one of the available clock sources by programming CLKCTRL[CLKSEL].

22.4.8 Counter clock prescaler

The EWM counter clock source can be prescaled by a clock divider, by programming CLKPRESCALER[CLK_DIV]. This divided clock is used to run the EWM counter.

NOTE

The divided clock used to run the EWM counter must be no more than half the frequency of the bus clock.

Chapter 23
Cyclic Redundancy Check (CRC)

23.1 Introduction

NOTE

For the chip-specific implementation details of this module's instances, see the chip configuration information.

The cyclic redundancy check (CRC) module generates 16/32-bit CRC code for error detection.

The CRC module provides a programmable polynomial, WAS, and other parameters required to implement a 16-bit or 32-bit CRC standard.

The 16/32-bit code is calculated for 32 bits of data at a time.

23.1.1 Features

Features of the CRC module include:

- Hardware CRC generator circuit using a 16-bit or 32-bit programmable shift register
- Programmable initial seed value and polynomial
- Option to transpose input data or output data (the CRC result) bitwise or bytewise. This option is required for certain CRC standards. A bytewise transpose operation is not possible when accessing the CRC data register via 8-bit accesses. In this case, the user's software must perform the bytewise transpose function.
- Option for inversion of final CRC result
- 32-bit CPU register programming interface

23.1.2 Block diagram

The following is a block diagram of the CRC.

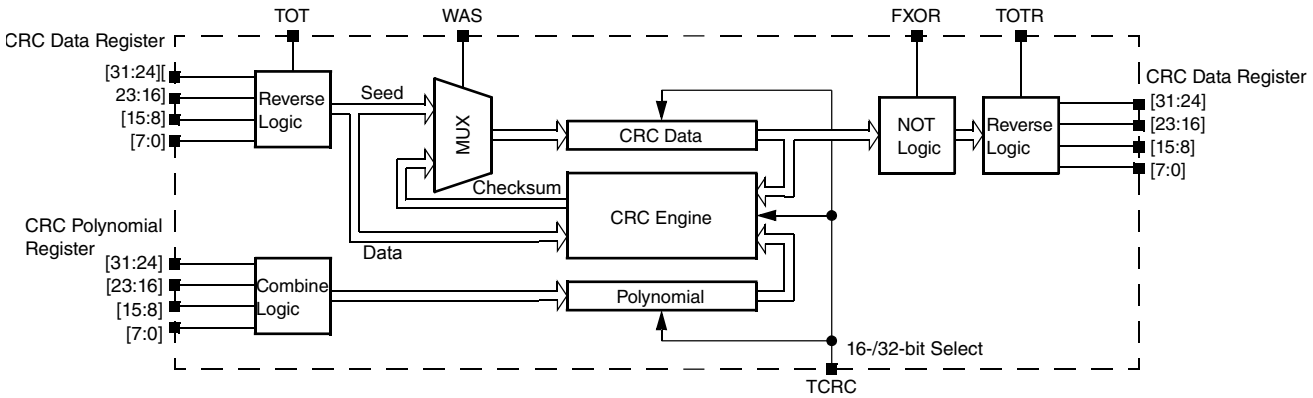


Figure 23-1. Programmable cyclic redundancy check (CRC) block diagram

23.1.3 Modes of operation

Various MCU modes affect the CRC module's functionality.

23.1.3.1 Run mode

This is the basic mode of operation.

23.1.3.2 Low-power modes (Wait or Stop)

Any CRC calculation in progress stops when the MCU enters a low-power mode that disables the module clock. It resumes after the clock is enabled or via the system reset for exiting the low-power mode. Clock gating for this module is dependent on the MCU.

23.2 Memory map and register descriptions

CRC memory map

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
E1C0	CRC Data register (CRC_DATA)	32	R/W	FFFF_FFFFh	23.2.1/459
E1C2	CRC Polynomial register (CRC_GPOLY)	32	R/W	0000_1021h	23.2.2/460
E1C4	CRC Control register (CRC_CTRL)	32	R/W	0000_0000h	23.2.3/461

23.2.1 CRC Data register (CRC_DATA)

The CRC Data register contains the value of the seed, data, and checksum. When CTRL[WAS] is set, any write to the data register is regarded as the seed value. When CTRL[WAS] is cleared, any write to the data register is regarded as data for general CRC computation.

In 16-bit CRC mode, the HU and HL fields are not used for programming the seed value, and reads of these fields return an indeterminate value. In 32-bit CRC mode, all fields are used for programming the seed value.

When programming data values, the values can be written 8 bits, 16 bits, or 32 bits at a time, provided all bytes are contiguous; with MSB of data value written first.

After all data values are written, the CRC result can be read from this data register. In 16-bit CRC mode, the CRC result is available in the LU and LL fields. In 32-bit CRC mode, all fields contain the result. Reads of this register at any time return the intermediate CRC value, provided the CRC module is configured.

NOTE

The address for this module on this chip is presented in terms on 16-bit words. However, in the case of an 8-bit write to this register, use its address in terms of bytes (which is double the value of the 16-bit word address).

Address: E1C0h base + 0h offset = E1C0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	HU								HL								LU								LL							
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

CRC_DATA field descriptions

Field	Description
31–24 HU	CRC High Upper Byte In 16-bit CRC mode (CTRL[TCRC] is 0), this field is not used for programming a seed value. In 32-bit CRC mode (CTRL[TCRC] is 1), values written to this field are part of the seed value when CTRL[WAS] is 1. When CTRL[WAS] is 0, data written to this field is used for CRC checksum generation in both 16-bit and 32-bit CRC modes.
23–16 HL	CRC High Lower Byte In 16-bit CRC mode (CTRL[TCRC] is 0), this field is not used for programming a seed value. In 32-bit CRC mode (CTRL[TCRC] is 1), values written to this field are part of the seed value when CTRL[WAS] is 1. When CTRL[WAS] is 0, data written to this field is used for CRC checksum generation in both 16-bit and 32-bit CRC modes.

Table continues on the next page...

CRC_DATA field descriptions (continued)

Field	Description
15–8 LU	CRC Low Upper Byte When CTRL[WAS] is 1, values written to this field are part of the seed value. When CTRL[WAS] is 0, data written to this field is used for CRC checksum generation.
LL	CRC Low Lower Byte When CTRL[WAS] is 1, values written to this field are part of the seed value. When CTRL[WAS] is 0, data written to this field is used for CRC checksum generation.

23.2.2 CRC Polynomial register (CRC_GPOLY)

This register contains the value of the polynomial for the CRC calculation. The HIGH field contains the upper 16 bits of the CRC polynomial, which are used only in 32-bit CRC mode. Writes to the HIGH field are ignored in 16-bit CRC mode. The LOW field contains the lower 16 bits of the CRC polynomial, which are used in both 16- and 32-bit CRC modes.

Address: E1C0h base + 2h offset = E1C2h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	HIGH																LOW															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	1	0	0	0	0	1

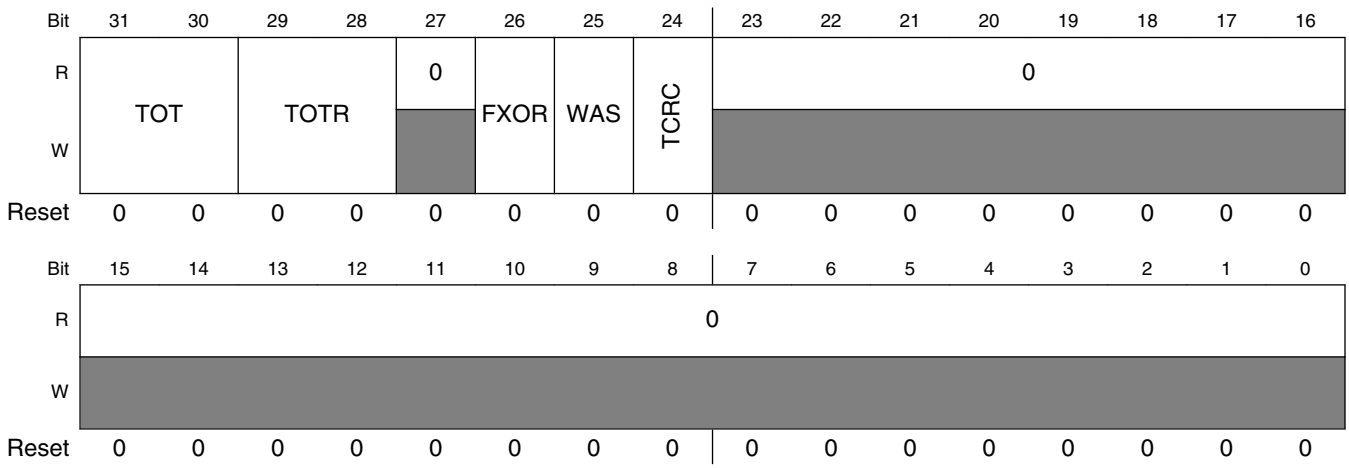
CRC_GPOLY field descriptions

Field	Description
31–16 HIGH	High Polynominal Half-word Writable and readable in 32-bit CRC mode (CTRL[TCRC] is 1). This field is not writable in 16-bit CRC mode (CTRL[TCRC] is 0).
LOW	Low Polynominal Half-word Writable and readable in both 32-bit and 16-bit CRC modes.

23.2.3 CRC Control register (CRC_CTRL)

This register controls the configuration and working of the CRC module. Appropriate bits must be set before starting a new CRC calculation. A new CRC calculation is initialized by asserting CTRL[WAS] and then writing the seed into the CRC data register.

Address: E1C0h base + 4h offset = E1C4h



CRC_CTRL field descriptions

Field	Description
31–30 TOT	Type Of Transpose For Writes Defines the transpose configuration of the data written to the CRC data register. See the description of the transpose feature for the available transpose options. 00 No transposition. 01 Bits in bytes are transposed; bytes are not transposed. 10 Both bits in bytes and bytes are transposed. 11 Only bytes are transposed; no bits in a byte are transposed.
29–28 TOTR	Type Of Transpose For Read Identifies the transpose configuration of the value read from the CRC Data register. See the description of the transpose feature for the available transpose options. 00 No transposition. 01 Bits in bytes are transposed; bytes are not transposed. 10 Both bits in bytes and bytes are transposed. 11 Only bytes are transposed; no bits in a byte are transposed.
27 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
26 FXOR	Complement Read Of CRC Data Register Some CRC protocols require the final checksum to be XORed with 0xFFFFFFFF or 0xFFFF. Asserting this bit enables on the fly complementing of read data.

Table continues on the next page...

CRC_CTRL field descriptions (continued)

Field	Description
	0 No XOR on reading. 1 Invert or complement the read value of the CRC Data register.
25 WAS	Write CRC Data Register As Seed When asserted, a value written to the CRC data register is considered a seed value. When deasserted, a value written to the CRC data register is taken as data for CRC computation. 0 Writes to the CRC data register are data values. 1 Writes to the CRC data register are seed values.
24 TCRC	Width of CRC protocol. 0 16-bit CRC protocol. 1 32-bit CRC protocol.
Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

23.3 Functional description

23.3.1 CRC initialization/reinitialization

To enable the CRC calculation, the user must program CRC_CTRL[WAS], CRC_GPOLY, necessary parameters for transposition and CRC result inversion in the applicable registers. Asserting CRC_CTRL[WAS] enables the programming of the seed value into the CRC_DATA register.

After a completed CRC calculation, the module can be reinitialized for a new CRC computation by reasserting CRC_CTRL[WAS] and programming a new, or previously used, seed value. All other parameters must be set before programming the seed value and subsequent data values.

23.3.2 CRC calculations

In 16-bit and 32-bit CRC modes, data values can be programmed 8 bits, 16 bits, or 32 bits at a time, provided all bytes are contiguous. Noncontiguous bytes can lead to an incorrect CRC computation.

23.3.2.1 16-bit CRC

To compute a 16-bit CRC:

1. Clear CRC_CTRL[TCRC] to enable 16-bit CRC mode.
2. Program the transpose and complement options in the CTRL register as required for the CRC calculation. See Transpose feature and CRC result complement for details.
3. Write a 16-bit polynomial to the CRC_GPOLY[LOW] field. The CRC_GPOLY[HIGH] field is not usable in 16-bit CRC mode.
4. Set CRC_CTRL[WAS] to program the seed value.
5. Write a 16-bit seed to CRC_DATA[LU:LL]. CRC_DATA[HU:HL] are not used.
6. Clear CRC_CTRL[WAS] to start writing data values.
7. Write data values into CRC_DATA[HU:HL:LU:LL]. A CRC is computed on every data value write, and the intermediate CRC result is stored back into CRC_DATA[LU:LL].
8. When all values have been written, read the final CRC result from CRC_DATA[LU:LL].

Transpose and complement operations are performed on the fly while reading or writing values. See Transpose feature and CRC result complement for details.

23.3.2.2 32-bit CRC

To compute a 32-bit CRC:

1. Set CRC_CTRL[TCRC] to enable 32-bit CRC mode.
2. Program the transpose and complement options in the CTRL register as required for the CRC calculation. See Transpose feature and CRC result complement for details.
3. Write a 32-bit polynomial to CRC_GPOLY[HIGH:LOW].
4. Set CRC_CTRL[WAS] to program the seed value.
5. Write a 32-bit seed to CRC_DATA[HU:HL:LU:LL].
6. Clear CRC_CTRL[WAS] to start writing data values.
7. Write data values into CRC_DATA[HU:HL:LU:LL]. A CRC is computed on every data value write, and the intermediate CRC result is stored back into CRC_DATA[HU:HL:LU:LL].
8. When all values have been written, read the final CRC result from CRC_DATA[HU:HL:LU:LL]. The CRC is calculated bytewise, and two clocks are required to complete one CRC calculation.

Transpose and complement operations are performed on the fly while reading or writing values. See Transpose feature and CRC result complement for details.

23.3.3 Transpose feature

By default, the transpose feature is not enabled. However, some CRC standards require the input data and/or the final checksum to be transposed. The user software has the option to configure each transpose operation separately, as desired by the CRC standard. The data is transposed on the fly while being read or written.

Some protocols use little endian format for the data stream to calculate a CRC. In this case, the transpose feature usefully flips the bits. This transpose option is one of the types supported by the CRC module.

23.3.3.1 Types of transpose

The CRC module provides several types of transpose functions to flip the bits and/or bytes, for both writing input data and reading the CRC result, separately using the CTRL[TOT] or CTRL[TOTR] fields, according to the CRC calculation being used.

The following types of transpose functions are available for writing to and reading from the CRC data register:

1. CTRL[TOT] or CTRL[TOTR] is 00.

   No transposition occurs.

2. CTRL[TOT] or CTRL[TOTR] is 01

   Bits in a byte are transposed, while bytes are not transposed.

   reg[31:0] becomes {reg[24:31], reg[16:23], reg[8:15], reg[0:7]}

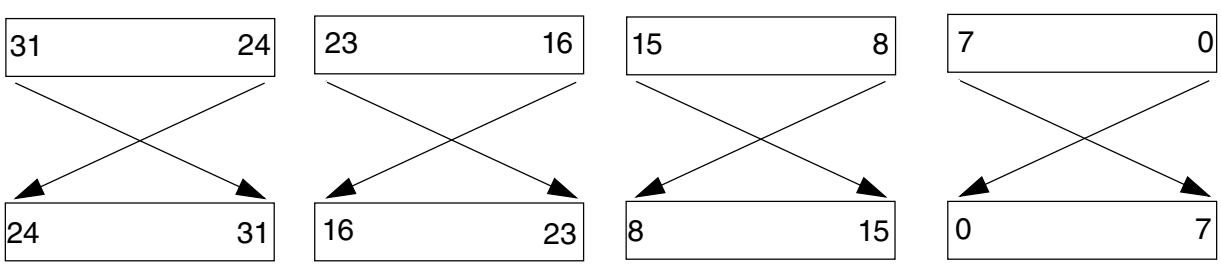


Figure 23-2. Transpose type 01

3. CTRL[TOT] or CTRL[TOTR] is 10.

   Both bits in bytes and bytes are transposed.

   reg[31:0] becomes = {reg[0:7], reg[8:15],reg[16:23], reg[24:31]}

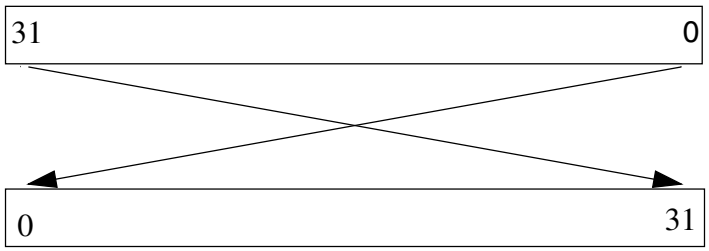


Figure 23-3. Transpose type 10

4. CTRL[TOT] or CTRL[TOTR] is 11.

   Bytes are transposed, but bits are not transposed.

   reg[31:0] becomes {reg[7:0], reg[15:8], reg[23:16], reg[31:24]}

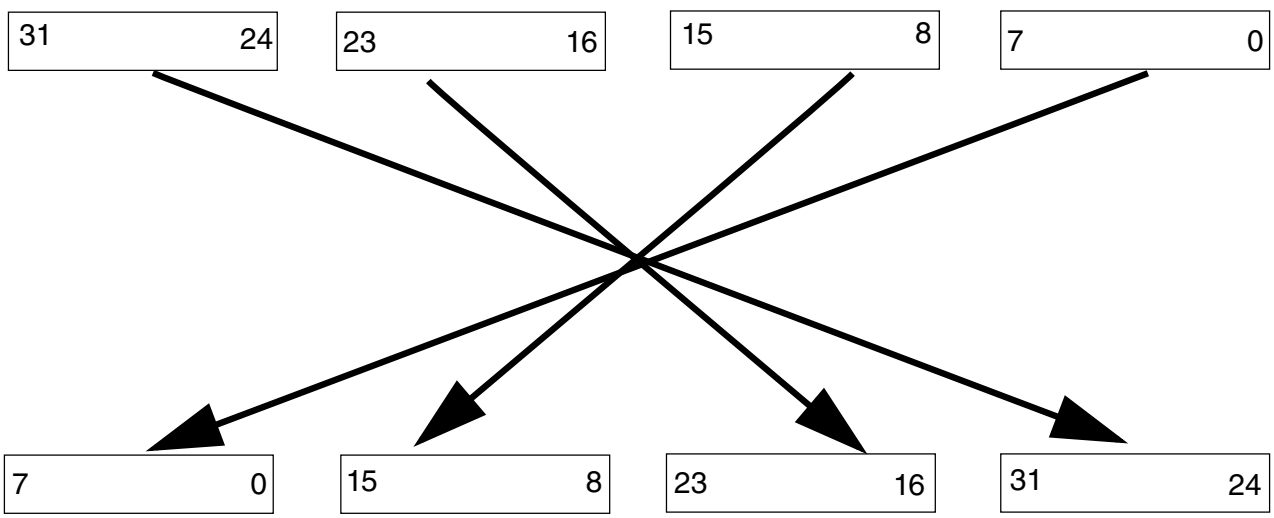


Figure 23-4. Transpose type 11

NOTE

For 8-bit and 16-bit write accesses to the CRC data register, the data is transposed with zeros on the unused byte or bytes (taking 32 bits as a whole), but the CRC is calculated on the valid byte(s) only. When reading the CRC data register for a 16-bit CRC result and using transpose options 10 and 11, the resulting value after transposition resides in the CRC[HU:HL] fields. The user software must account for this situation when reading the 16-bit CRC result, so reading 32 bits is preferred.

23.3.4 CRC result complement

When CTRL[FXOR] is set, the checksum is complemented. The CRC result complement function outputs the complement of the checksum value stored in the CRC data register every time the CRC data register is read. When CTRL[FXOR] is cleared, reading the CRC data register accesses the raw checksum value.

Chapter 24
16-bit SAR Analog-to-Digital Converter (ADC16)

24.1 Introduction

NOTE

For the chip-specific implementation details of this module's instances, see the chip configuration information.

The 16-bit analog-to-digital converter (ADC) is a successive approximation ADC designed for operation within an integrated microcontroller system-on-chip.

NOTE

For the chip specific modes of operation, see the power management information of the device.

24.1.1 Features

Following are the features of the ADC module.

- Linear successive approximation algorithm with up to 16-bit resolution
- Up to 24 single-ended external analog inputs
- Output modes:
  - single-ended 16-bit, 12-bit, 10-bit, and 8-bit modes
- Output in right-justified unsigned format for single-ended
- Single or continuous conversion, that is, automatic return to idle after single conversion
- Configurable sample time and conversion speed/power
- Conversion complete/hardware average complete flag and interrupt

- Input clock selectable from up to four sources
- Operation in low-power modes for lower noise
- Asynchronous clock source for lower noise operation with option to output the clock
- Selectable hardware conversion trigger with hardware channel select
- Automatic compare with interrupt for less-than, greater-than or equal-to, within range, or out-of-range, programmable value
- Temperature sensor
- Hardware average function
- Selectable voltage reference: external or alternate
- Self-Calibration mode

24.1.2 Block diagram

The following figure is the ADC module block diagram.

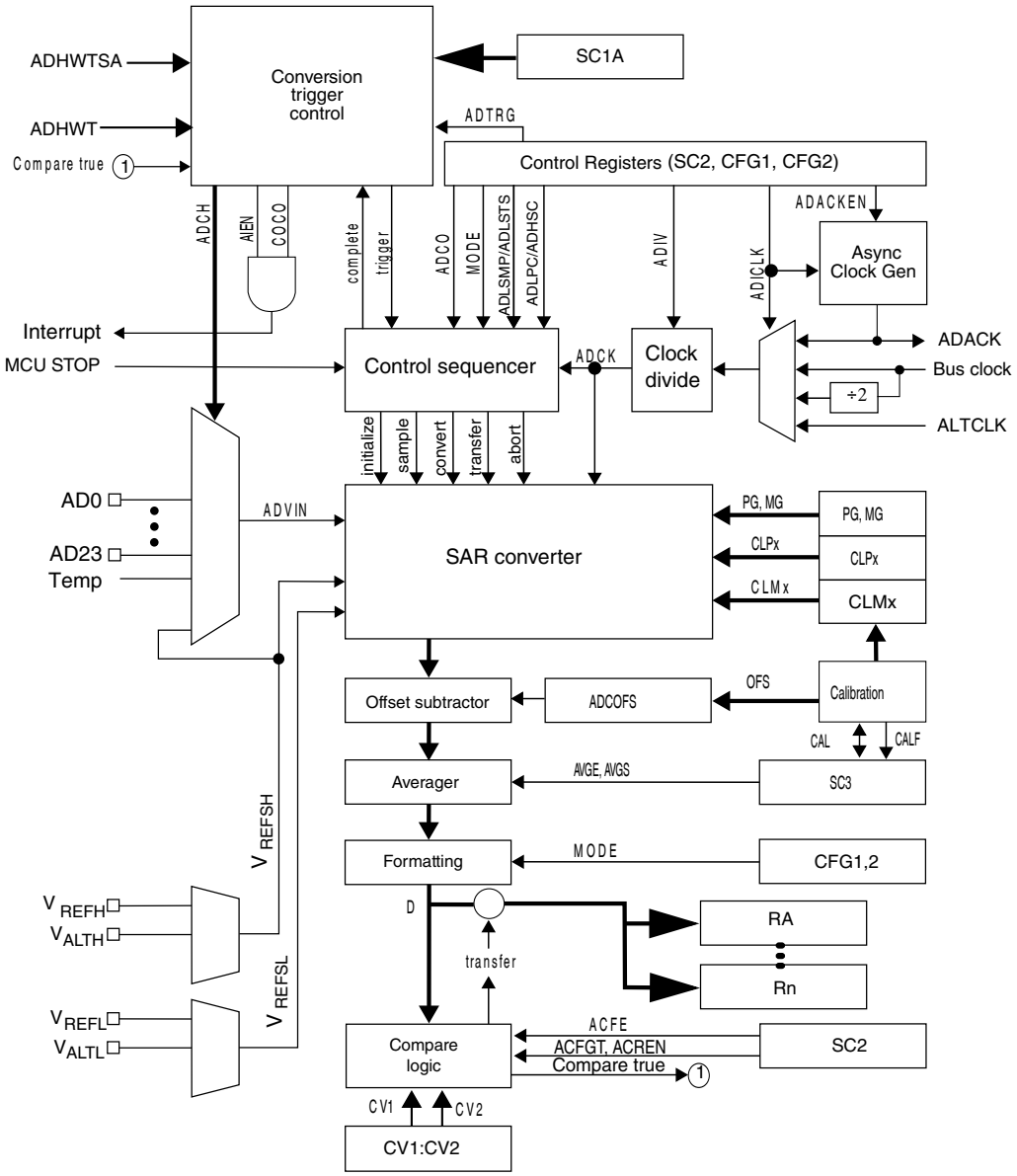


Figure 24-1. ADC block diagram

24.2 ADC signal descriptions

The ADC module supports up to 24 single-ended inputs.

The ADC also requires four supply/reference/ground connections.

NOTE

Refer to ADC configuration section in chip configuration chapter for the number of channels supported on this device.

Table 24-1. ADC signal descriptions

Signal	Description	I/O
AD*n*	Single-Ended Analog Channel Inputs	I
V_{REFSH}	Voltage Reference Select High	I
V_{REFSL}	Voltage Reference Select Low	I
V_{DDA}	Analog Power Supply	I
V_{SSA}	Analog Ground	I

24.2.1 Analog Power (V_{DDA})

The ADC analog portion uses V_{DDA} as its power connection. In some packages, V_{DDA} is connected internally to V_{DD}. If externally available, connect the V_{DDA} pin to the same voltage potential as V_{DD}. External filtering may be necessary to ensure clean V_{DDA} for good results.

24.2.2 Analog Ground (V_{SSA})

The ADC analog portion uses V_{SSA} as its ground connection. In some packages, V_{SSA} is connected internally to V_{SS}. If externally available, connect the V_{SSA} pin to the same voltage potential as V_{SS}.

24.2.3 Voltage Reference Select

V_{REFSH} and V_{REFSL} are the high and low reference voltages for the ADC module.

The ADC can be configured to accept one of two voltage reference pairs for V_{REFSH} and V_{REFSL}. Each pair contains a positive reference that must be between the minimum Ref Voltage High and V_{DDA}, and a ground reference that must be at the same potential as V_{SSA}. The two pairs are external (V_{REFH} and V_{REFL}) and alternate (V_{ALTH} and V_{ALTL}). These voltage references are selected using SC2[REFSEL]. The alternate V_{ALTH} and V_{ALTL} voltage reference pair may select additional external pins or internal sources depending on MCU configuration. See the chip configuration information on the Voltage References specific to this MCU.

In some packages, V_{REFH} is connected in the package to V_{DDA} and V_{REFL} to V_{SSA}. If externally available, the positive reference(s) may be connected to the same potential as V_{DDA} or may be driven by an external source to a level between the minimum Ref Voltage High and the V_{DDA} potential. V_{REFH} must never exceed V_{DDA}. Connect the ground references to the same voltage potential as V_{SSA}.

24.2.4 Analog Channel Inputs (ADx)

The ADC module supports up to 24 single-ended analog inputs. A single-ended input is selected for conversion through the SC1[ADCH] channel select bits.

24.3 Memory map and register definitions

This section describes the ADC registers.

ADC memory map

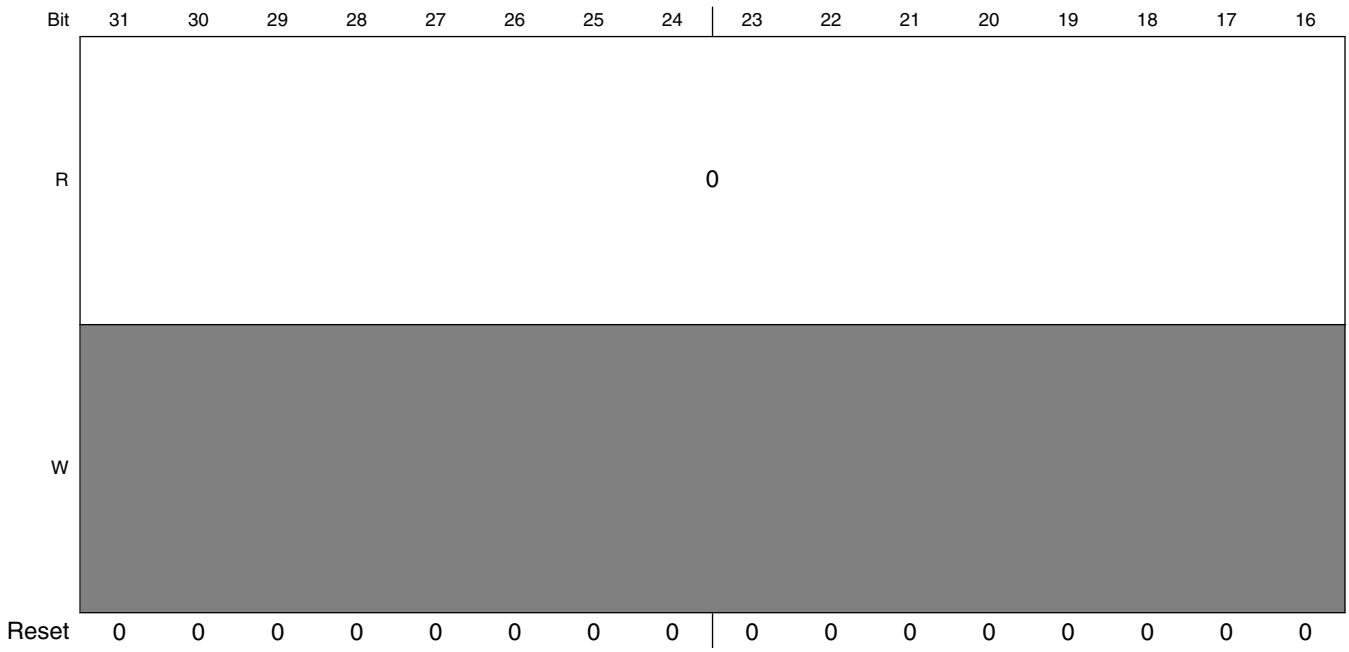
Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
E580	ADC Status and Control Registers 1 (ADC16_SC1A)	32	R/W	0000_001Fh	24.3.1/472
E584	ADC Configuration Register 1 (ADC16_CFG1)	32	R/W	0000_0000h	24.3.2/475
E586	ADC Configuration Register 2 (ADC16_CFG2)	32	R/W	0000_0000h	24.3.3/477
E588	ADC Data Result Register (ADC16_RA)	32	R	0000_0000h	24.3.4/478
E58C	Compare Value Registers (ADC16_CV1)	32	R/W	0000_0000h	24.3.5/479
E58E	Compare Value Registers (ADC16_CV2)	32	R/W	0000_0000h	24.3.5/479
E590	Status and Control Register 2 (ADC16_SC2)	32	R/W	0000_0000h	24.3.6/480
E592	Status and Control Register 3 (ADC16_SC3)	32	R/W	0000_0000h	24.3.7/482
E594	ADC Offset Correction Register (ADC16_OFS)	32	R/W	0000_0004h	24.3.8/484
E596	ADC Plus-Side Gain Register (ADC16_PG)	32	R/W	0000_8200h	24.3.9/484
E59A	ADC Plus-Side General Calibration Value Register (ADC16_CLPD)	32	R/W	0000_000Ah	24.3.10/485
E59C	ADC Plus-Side General Calibration Value Register (ADC16_CLPS)	32	R/W	0000_0020h	24.3.11/485
E59E	ADC Plus-Side General Calibration Value Register (ADC16_CLP4)	32	R/W	0000_0200h	24.3.12/486
E5A0	ADC Plus-Side General Calibration Value Register (ADC16_CLP3)	32	R/W	0000_0100h	24.3.13/486
E5A2	ADC Plus-Side General Calibration Value Register (ADC16_CLP2)	32	R/W	0000_0080h	24.3.14/487
E5A4	ADC Plus-Side General Calibration Value Register (ADC16_CLP1)	32	R/W	0000_0040h	24.3.15/487
E5A6	ADC Plus-Side General Calibration Value Register (ADC16_CLP0)	32	R/W	0000_0020h	24.3.16/488

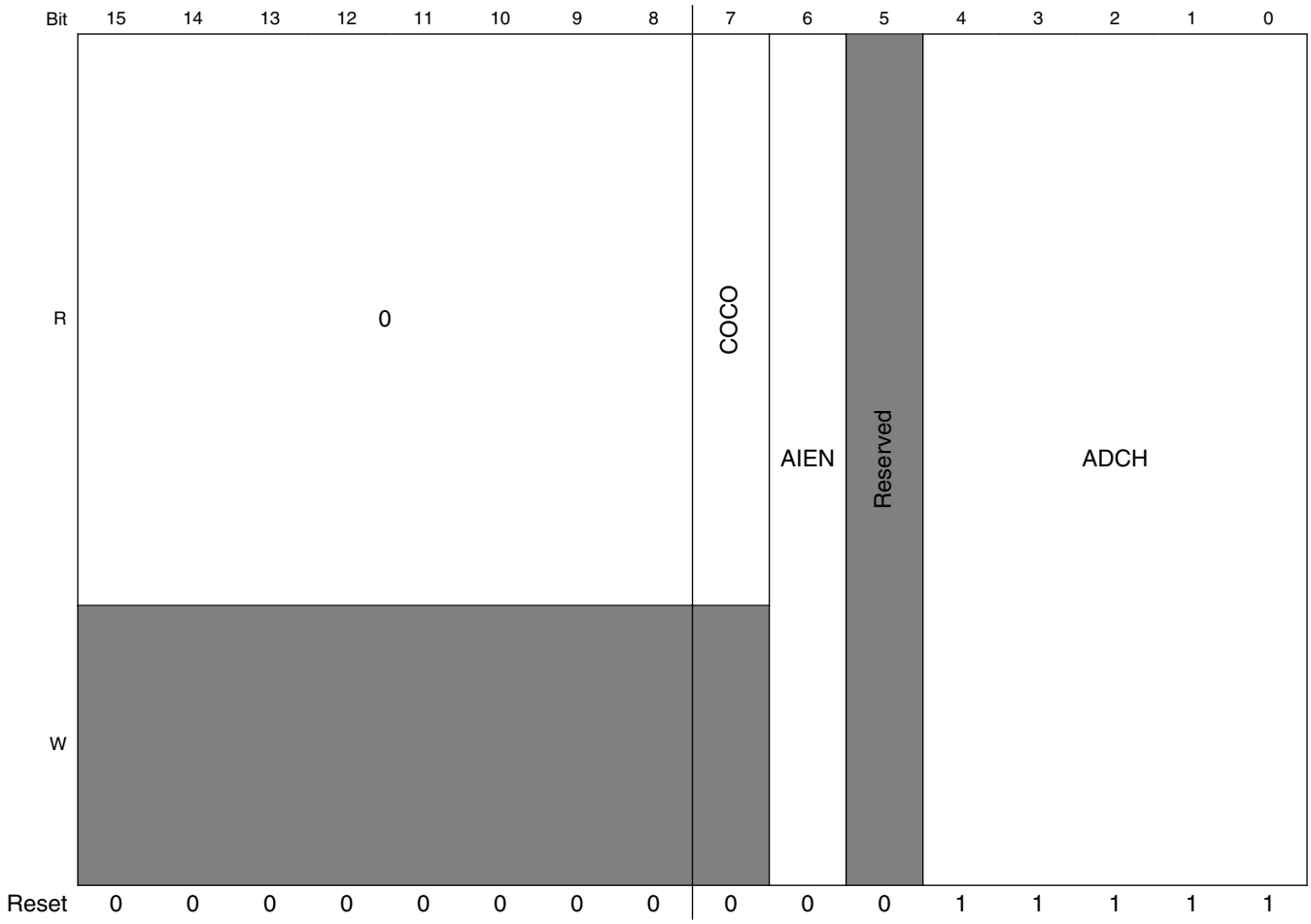
24.3.1 ADC Status and Control Registers 1 (ADC*x*_SC1*n*)

SC1A is used for both software and hardware trigger modes of operation.

Writing SC1A while SC1A is actively controlling a conversion aborts the current conversion. In Software Trigger mode, when SC2[ADTRG]=0, writes to SC1A subsequently initiate a new conversion, if SC1[ADCH] contains a value other than all 1s.

Address: Base address + 0h offset + (2d × i), where i=0d to 0d





ADCx_SC1n field descriptions

Field	Description
31–8 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
7 COCO	Conversion Complete Flag This is a read-only field that is set each time a conversion is completed when the compare function is disabled, or SC2[ACFE]=0 and the hardware average function is disabled, or SC3[AVGE]=0. When the compare function is enabled, or SC2[ACFE]=1, COCO is set upon completion of a conversion only if the compare result is true. When the hardware average function is enabled, or SC3[AVGE]=1, COCO is set upon completion of the selected number of conversions (determined by AVGS). COCO in SC1A is also set at the completion of a calibration sequence. COCO is cleared when the respective SC1n register is written or when the respective Rn register is read. 0 Conversion is not completed. 1 Conversion is completed.
6 AIEN	Interrupt Enable Enables conversion complete interrupts. When COCO becomes set while the respective AIEN is high, an interrupt is asserted. 0 Conversion complete interrupt is disabled. 1 Conversion complete interrupt is enabled.

Table continues on the next page...

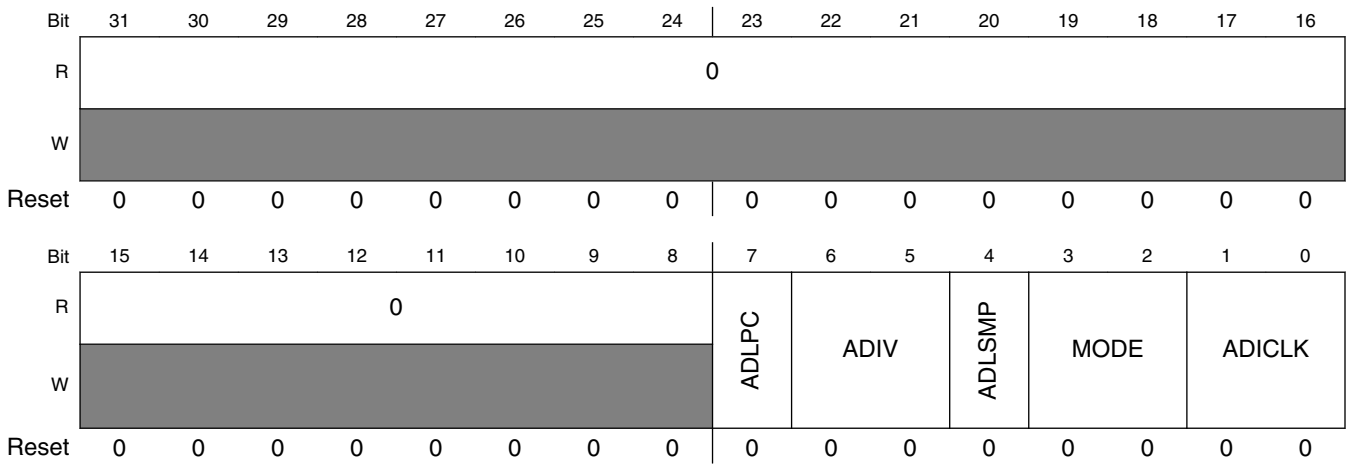
ADC*x*_SC1*n* field descriptions (continued)

Field	Description
5 Reserved	This field is reserved. This reserved bit should not be changed.
ADCH	Input channel select Selects one of the input channels. **NOTE:** Some of the input channel options in the bitfield-setting descriptions might not be available for your device. For the actual ADC channel assignments for your device, see the Chip Configuration details. The successive approximation converter subsystem is turned off when the channel select bits are all set, that is, ADCH = 11111. This feature allows explicit disabling of the ADC and isolation of the input channel from all sources. Terminating continuous conversions this way prevents an additional single conversion from being performed. It is not necessary to set ADCH to all 1s to place the ADC in a low-power state when continuous conversions are not enabled because the module automatically enters a low-power state when a conversion completes. 00000 AD0 is selected as input. 00001 AD1 is selected as input. 00010 AD2 is selected as input. 00011 AD3 is selected as input. 00100 AD4 is selected as input. 00101 AD5 is selected as input. 00110 AD6 is selected as input. 00111 AD7 is selected as input. 01000 AD8 is selected as input. 01001 AD9 is selected as input. 01010 AD10 is selected as input. 01011 AD11 is selected as input. 01100 AD12 is selected as input. 01101 AD13 is selected as input. 01110 AD14 is selected as input. 01111 AD15 is selected as input. 10000 AD16 is selected as input. 10001 AD17 is selected as input. 10010 AD18 is selected as input. 10011 AD19 is selected as input. 10100 AD20 is selected as input. 10101 AD21 is selected as input. 10110 AD22 is selected as input. 10111 AD23 is selected as input. 11000 Reserved. 11001 Reserved. 11010 Temp Sensor (single-ended) is selected as input. 11011 Bandgap (single-ended) is selected as input. 11100 Reserved. 11101 V_{REFSH} is selected as input. Voltage reference selected is determined by SC2[REFSEL]. 11110 V_{REFSL} is selected as input. Voltage reference selected is determined by SC2[REFSEL]. 11111 Module is disabled.

24.3.2 ADC Configuration Register 1 (ADCx_CFG1)

The configuration Register 1 (CFG1) selects the mode of operation, clock source, clock divide, and configuration for low power or long sample time.

Address: Base address + 4h offset



ADCx_CFG1 field descriptions

Field	Description
31–8 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
7 ADLPC	Low-Power Configuration Controls the power configuration of the successive approximation converter. This optimizes power consumption when higher sample rates are not required. 0 Normal power configuration. 1 Low-power configuration. The power is reduced at the expense of maximum clock speed.
6–5 ADIV	Clock Divide Select Selects the divide ratio used by the ADC to generate the internal clock ADCK. 00 The divide ratio is 1 and the clock rate is input clock. 01 The divide ratio is 2 and the clock rate is (input clock)/2. 10 The divide ratio is 4 and the clock rate is (input clock)/4. 11 The divide ratio is 8 and the clock rate is (input clock)/8.
4 ADLSMP	Sample Time Configuration Selects between different sample times based on the conversion mode selected. This field adjusts the sample period to allow higher impedance inputs to be accurately sampled or to maximize conversion speed for lower impedance inputs. Longer sample times can also be used to lower overall power consumption if continuous conversions are enabled and high conversion rates are not required. When ADLSMP=1, the long sample time select bits, (ADLSTS[1:0]), can select the extent of the long sample time.

Table continues on the next page...

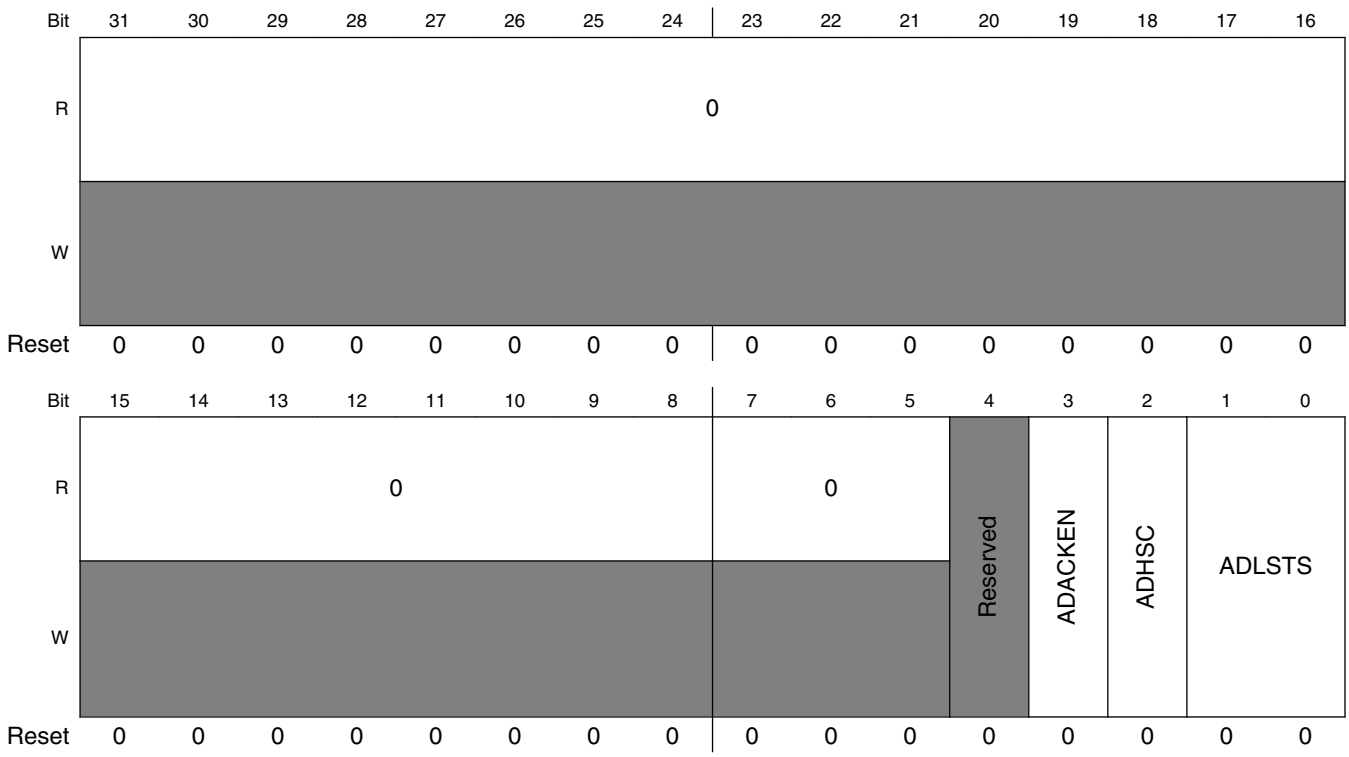
ADCx_CFG1 field descriptions (continued)

Field	Description
	0 Short sample time. 1 Long sample time.
3–2 MODE	Conversion mode selection Selects the ADC resolution mode. 00 It is single-ended 8-bit conversion. 01 It is single-ended 12-bit conversion . 10 It is single-ended 10-bit conversion. 11 It is single-ended 16-bit conversion..
ADICLK	Input Clock Select Selects the input clock source to generate the internal clock, ADCK. Note that when the ADACK clock source is selected, it is not required to be active prior to conversion start. When it is selected and it is not active prior to a conversion start, when CFG2[ADACKEN]=0, the asynchronous clock is activated at the start of a conversion and deactivated when conversions are terminated. In this case, there is an associated clock startup delay each time the clock source is re-activated. 00 Bus clock 01 Bus clock divided by 2(BUSCLK/DIV2) 10 Alternate clock (ALTCLK) 11 Asynchronous clock (ADACK)

24.3.3 ADC Configuration Register 2 (ADCx_CFG2)

Configuration Register 2 (CFG2) selects the special high-speed configuration for very high speed conversions and selects the long sample time duration during long sample mode.

Address: Base address + 6h offset



ADCx_CFG2 field descriptions

Field	Description
31–8 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
7–5 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
4 Reserved	This field is reserved.
3 ADACKEN	Asynchronous Clock Output Enable Enables the asynchronous clock source and the clock source output regardless of the conversion and status of CFG1[ADICLK]. Based on MCU configuration, the asynchronous clock may be used by other modules. See chip configuration information. Setting this field allows the clock to be used even while the ADC is idle or operating from a different clock source. Also, latency of initiating a single or first-continuous conversion with the asynchronous clock selected is reduced because the ADACK clock is already operational.

Table continues on the next page...

ADCx_CFG2 field descriptions (continued)

Field	Description
	0 Asynchronous clock output disabled; Asynchronous clock is enabled only if selected by ADICLK and a conversion is active. 1 Asynchronous clock and clock output is enabled regardless of the state of the ADC.
2 ADHSC	High-Speed Configuration Configures the ADC for very high-speed operation. The conversion sequence is altered with 2 ADCK cycles added to the conversion time to allow higher speed conversion clocks. 0 Normal conversion sequence selected. 1 High-speed conversion sequence selected with 2 additional ADCK cycles to total conversion time.
ADLSTS	Long Sample Time Select Selects between the extended sample times when long sample time is selected, that is, when CFG1[ADLSMP]=1. This allows higher impedance inputs to be accurately sampled or to maximize conversion speed for lower impedance inputs. Longer sample times can also be used to lower overall power consumption when continuous conversions are enabled if high conversion rates are not required. 00 Default longest sample time; 20 extra ADCK cycles; 24 ADCK cycles total. 01 12 extra ADCK cycles; 16 ADCK cycles total sample time. 10 6 extra ADCK cycles; 10 ADCK cycles total sample time. 11 2 extra ADCK cycles; 6 ADCK cycles total sample time.

24.3.4 ADC Data Result Register (ADCx_R*n*)

The data result (RA) register contains the result of an ADC conversion of the channel selected by the status and channel control register (SC1A).

The following table describes the behavior of the data result registers in the different modes of operation.

Table 24-2. Data result register description

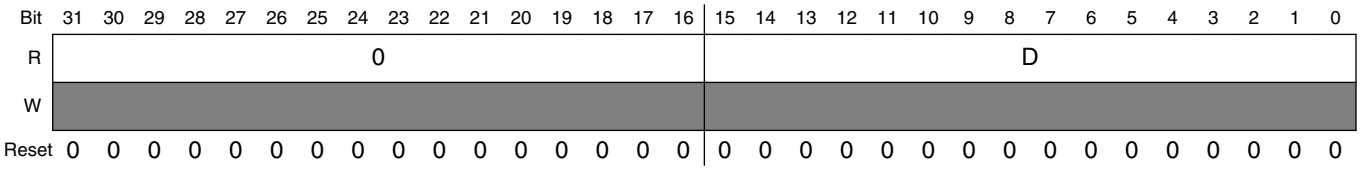
Conversion mode	D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0	Format
16-bit single-ended	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	Unsigned right justified
12-bit single-ended	0	0	0	0	D	D	D	D	D	D	D	D	D	D	D	D	Unsigned right-justified
10-bit single-ended	0	0	0	0	0	0	D	D	D	D	D	D	D	D	D	D	Unsigned right-justified
8-bit single-ended	0	0	0	0	0	0	0	0	D	D	D	D	D	D	D	D	Unsigned right-justified

NOTE

S: Sign bit or sign bit extension;

D: Data, which is 2's complement data if indicated

Address: Base address + 8h offset + (2d × i), where i=0d to 0d



ADCx_Rn field descriptions

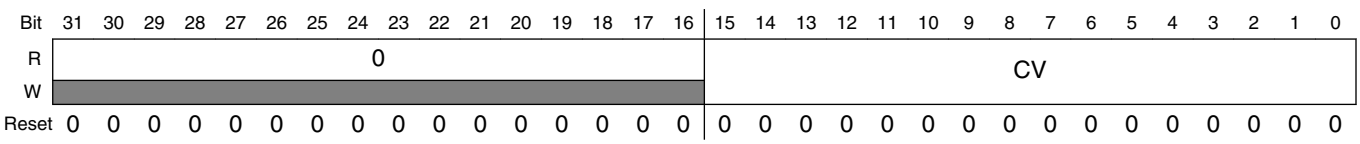
Field	Description
31–16 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
D	Data result

24.3.5 Compare Value Registers (ADCx_CVn)

The Compare Value Registers (CV1 and CV2) contain a compare value used to compare the conversion result when the compare function is enabled, that is, SC2[ACFE]=1. This register is formatted in the same way as the Rn registers in different modes of operation for both bit position definition and value format using unsigned or sign-extended 2's complement. Therefore, the compare function uses only the CVn fields that are related to the ADC mode of operation.

The compare value 2 register (CV2) is used only when the compare range function is enabled, that is, SC2[ACREN]=1.

Address: Base address + Ch offset + (2d × i), where i=0d to 1d



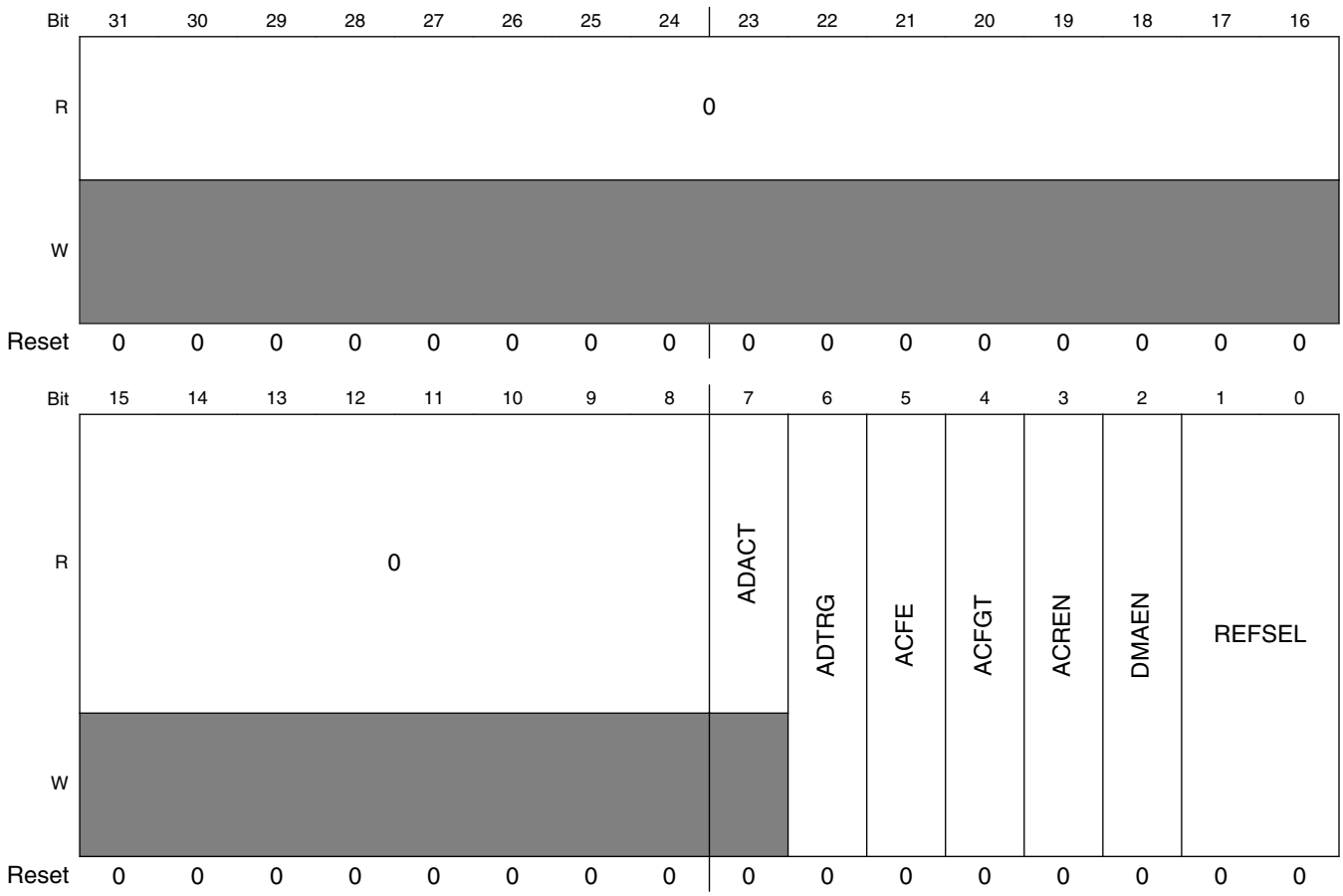
ADCx_CVn field descriptions

Field	Description
31–16 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
CV	Compare Value.

24.3.6 Status and Control Register 2 (ADCx_SC2)

The status and control register 2 (SC2) contains the conversion active, hardware/software trigger select, compare function, and voltage reference select of the ADC module.

Address: Base address + 10h offset



ADCx_SC2 field descriptions

Field	Description
31–8 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
7 ADACT	Conversion Active Indicates that a conversion or hardware averaging is in progress. ADACT is set when a conversion is initiated and cleared when a conversion is completed or aborted. 0 Conversion not in progress. 1 Conversion in progress.
6 ADTRG	Conversion Trigger Select Selects the type of trigger used for initiating a conversion. Two types of trigger are selectable:

Table continues on the next page...

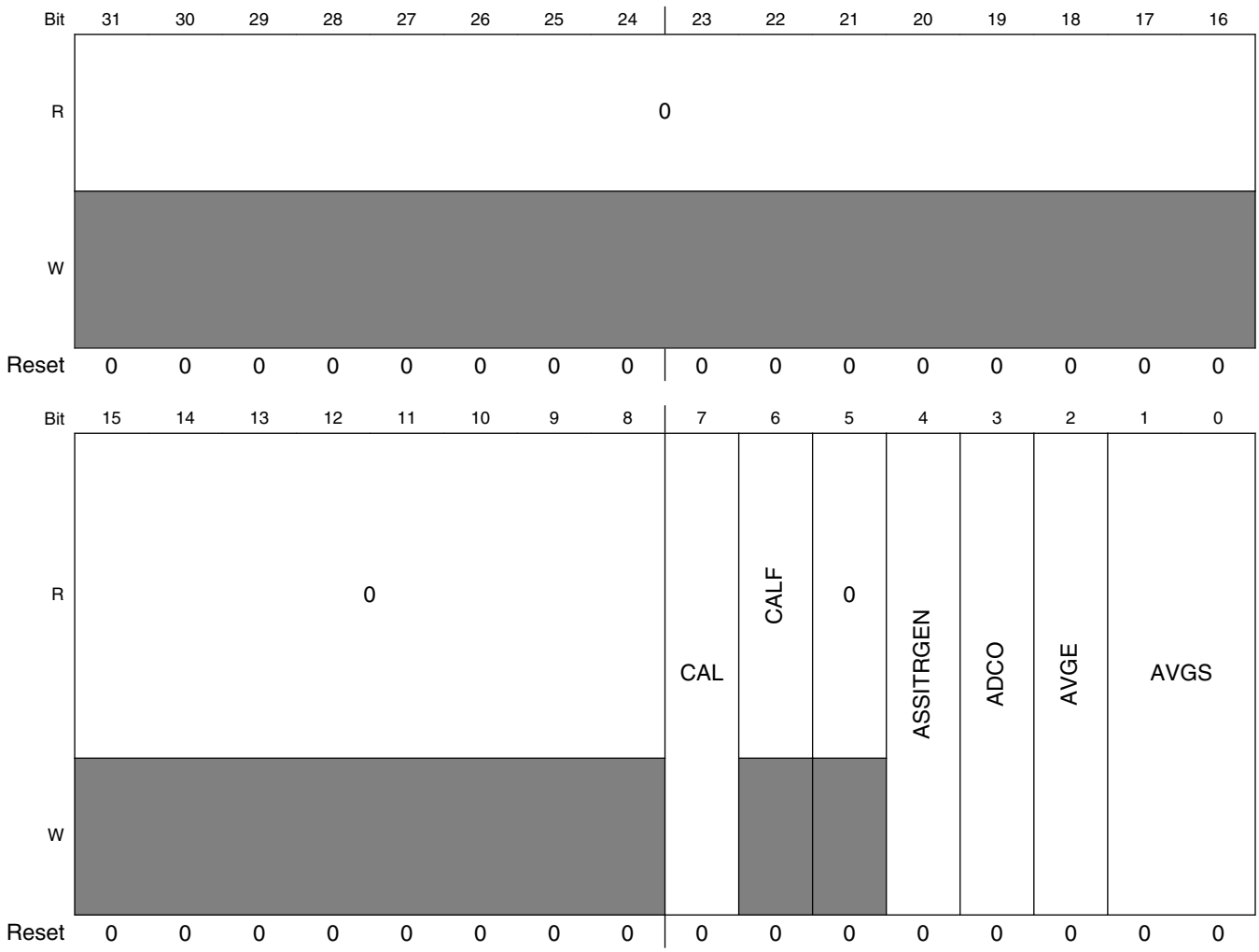
ADCx_SC2 field descriptions (continued)

Field	Description
	• Software trigger: When software trigger is selected, a conversion is initiated following a write to SC1A. • Hardware trigger: When hardware trigger is selected, a conversion is initiated following the assertion of the ADHWT input after a pulse of the ADHWTSn input. 0 Software trigger selected. 1 Hardware trigger selected.
5 ACFE	Compare Function Enable Enables the compare function. 0 Compare function disabled. 1 Compare function enabled.
4 ACFGT	Compare Function Greater Than Enable Configures the compare function to check the conversion result relative to the CV1 and CV2 based upon the value of ACREN. ACFE must be set for ACFGT to have any effect. 0 Configures less than threshold, outside range not inclusive and inside range not inclusive; functionality based on the values placed in CV1 and CV2. 1 Configures greater than or equal to threshold, outside and inside ranges inclusive; functionality based on the values placed in CV1 and CV2.
3 ACREN	Compare Function Range Enable Configures the compare function to check if the conversion result of the input being monitored is either between or outside the range formed by CV1 and CV2 determined by the value of ACFGT. ACFE must be set for ACFGT to have any effect. 0 Range function disabled. Only CV1 is compared. 1 Range function enabled. Both CV1 and CV2 are compared.
2 DMAEN	DMA Enable 0 DMA is disabled. 1 DMA is enabled and will assert the ADC DMA request during an ADC conversion complete event noted when any of the SC1n[COCO] flags is asserted.
REFSEL	Voltage Reference Selection Selects the voltage reference source used for conversions. 00 Default voltage reference pin pair, that is, external pins V_{REFH} and V_{REFL} 01 Alternate reference pair, that is, V_{ALTH} and V_{ALTL} . This pair may be additional external pins or internal sources depending on the MCU configuration. See the chip configuration information for details specific to this MCU 10 Internal bandgap reference and associated ground reference (V_{BGH} and V_{BGL}). Consult the Chip Configuration information for details specific to this MCU. 11 Reserved

24.3.7 Status and Control Register 3 (ADCx_SC3)

The Status and Control Register 3 (SC3) controls the calibration, continuous convert, and hardware averaging functions of the ADC module.

Address: Base address + 12h offset



ADCx_SC3 field descriptions

Field	Description
31–8 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
7 CAL	Calibration Begins the calibration sequence when set. This field stays set while the calibration is in progress and is cleared when the calibration sequence is completed. CALF must be checked to determine the result of the calibration sequence. Once started, the calibration routine cannot be interrupted by writes to the ADC registers or the results will be invalid and CALF will set. Setting CAL will abort any current conversion.

Table continues on the next page...

ADCx_SC3 field descriptions (continued)

Field	Description
6 CALF	Calibration Failed Flag Displays the result of the calibration sequence. The calibration sequence will fail if SC2[ADTRG] = 1, any ADC register is written, or any stop mode is entered before the calibration sequence completes. Writing 1 to CALF clears it. 0 Calibration completed normally. 1 Calibration failed. ADC accuracy specifications are not guaranteed.
5 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
4 ASSITRGEN	Assist Trigger Enable Enables writes to ADCSC1 COCO bit to be reflected on ADTRG register bit. 0 Writes to ADCSC1 COCO bit don't have an affect on ADTRG. 1 Writes to ADCSC1 COCO bit will be reflected into ADTRG register. Note: When ASSITRGEN is set, writes to ADCSC1 are delayed by 1/2 bus cycle to allow ADCSC1 COCO write to be updated in ADTRG register so conversion type can be correctly generated (software if coco write is 1'b0 causing ADTRG to clear or hardware type if coco write is 1'b1.). Note: User must ensure no hardware trigger is generated between the time ADCSC1 COCO bit is written if value of ADTRG will change to guarantee correct conversion type is generated.
3 ADCO	Continuous Conversion Enable Enables continuous conversions. 0 One conversion or one set of conversions if the hardware average function is enabled, that is, AVGE=1, after initiating a conversion. 1 Continuous conversions or sets of conversions if the hardware average function is enabled, that is, AVGE=1, after initiating a conversion.
2 AVGE	Hardware Average Enable Enables the hardware average function of the ADC. 0 Hardware average function disabled. 1 Hardware average function enabled.
AVGS	Hardware Average Select Determines how many ADC conversions will be averaged to create the ADC average result. 00 4 samples averaged. 01 8 samples averaged. 10 16 samples averaged. 11 32 samples averaged.

24.3.8 ADC Offset Correction Register (ADC*x*_OFS)

The ADC Offset Correction Register (OFS) contains the user-selected or calibration-generated offset error correction value. This register is a 2's complement, left-justified, 16-bit value . The value in OFS is subtracted from the conversion and the result is transferred into the result registers, Rn. If the result is greater than the maximum or less than the minimum result value, it is forced to the appropriate limit for the current mode of operation.

Address: Base address + 14h offset

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0																OFS															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0

ADC*x*_OFS field descriptions

Field	Description
31–16 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
OFS	Offset Error Correction Value

24.3.9 ADC Plus-Side Gain Register (ADC*x*_PG)

The Plus-Side Gain Register (PG) contains the gain error correction for the overall conversion in single-ended mode. PG, a 16-bit real number in binary format, is the gain adjustment factor, with the radix point fixed between ADPG15 and ADPG14. This register must be written by the user with the value described in the calibration procedure. Otherwise, the gain error specifications may not be met.

Address: Base address + 16h offset

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0																PG															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0

ADCx_PG field descriptions

Field	Description
31–16 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
PG	Plus-Side Gain

24.3.10 ADC Plus-Side General Calibration Value Register (ADCx_CLPD)

The Plus-Side General Calibration Value Registers (CLPx) contain calibration information that is generated by the calibration function. These registers contain seven calibration values of varying widths: CLP0[5:0], CLP1[6:0], CLP2[7:0], CLP3[8:0], CLP4[9:0], CLPS[5:0], and CLPD[5:0]. CLPx are automatically set when the self-calibration sequence is done, that is, CAL is cleared. If these registers are written by the user after calibration, the linearity error specifications may not be met.

Address: Base address + 1Ah offset

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0																										CLPD					
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0

ADCx_CLPD field descriptions

Field	Description
31–6 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
CLPD	Calibration Value Calibration Value

24.3.11 ADC Plus-Side General Calibration Value Register (ADCx_CLPS)

For more information, see CLPD register description.

Address: Base address + 1Ch offset

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0																										CLPS					
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0

ADCx_CLPS field descriptions

Field	Description
31–6 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
CLPS	Calibration Value Calibration Value

24.3.12 ADC Plus-Side General Calibration Value Register (ADCx_CLP4)

For more information, see CLPD register description.

Address: Base address + 1Eh offset

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0																						CLP4									
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0

ADCx_CLP4 field descriptions

Field	Description
31–10 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
CLP4	Calibration Value Calibration Value

24.3.13 ADC Plus-Side General Calibration Value Register (ADCx_CLP3)

For more information, see CLPD register description.

Address: Base address + 20h offset

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0																							CLP3								
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0

ADCx_CLP3 field descriptions

Field	Description
31–9 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

Table continues on the next page...

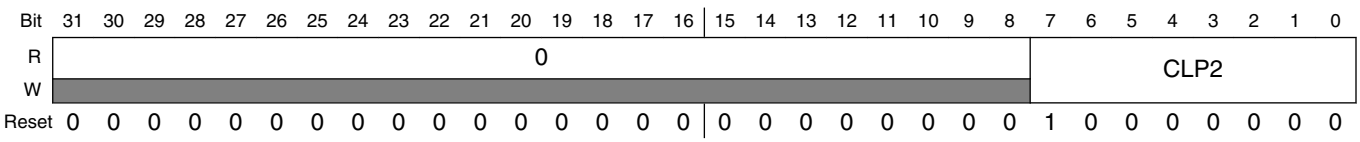
ADCx_CLP3 field descriptions (continued)

Field	Description
CLP3	Calibration Value Calibration Value

24.3.14 ADC Plus-Side General Calibration Value Register (ADCx_CLP2)

For more information, see CLPD register description.

Address: Base address + 22h offset



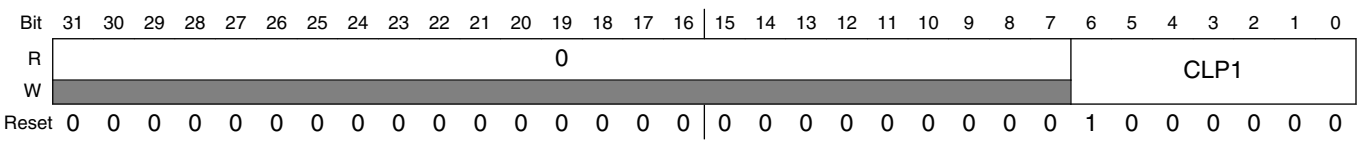
ADCx_CLP2 field descriptions

Field	Description
31–8 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
CLP2	Calibration Value Calibration Value

24.3.15 ADC Plus-Side General Calibration Value Register (ADCx_CLP1)

For more information, see CLPD register description.

Address: Base address + 24h offset



ADCx_CLP1 field descriptions

Field	Description
31–7 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
CLP1	Calibration Value Calibration Value

24.3.16 ADC Plus-Side General Calibration Value Register (ADCx_CLP0)

For more information, see CLPD register description.

Address: Base address + 26h offset

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0																CLP0															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0

ADCx_CLP0 field descriptions

Field	Description
31–6 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
CLP0	Calibration Value Calibration Value

24.4 Functional description

The ADC module is disabled during reset, in Low-Power Stop mode, or when SC1n[ADCH] are all high; see the power management information for details. The module is idle when a conversion has completed and another conversion has not been initiated. When it is idle and the asynchronous clock output enable is disabled, or CFG2[ADACKEN]= 0, the module is in its lowest power state. The ADC can perform an analog-to-digital conversion on any of the software selectable channels. All modes perform conversion by a successive approximation algorithm.

To meet accuracy specifications, the ADC module must be calibrated using the on-chip calibration function.

See Calibration function for details on how to perform calibration.

When the conversion is completed, the result is placed in the Rn data registers. The respective SC1n[COCO] is then set and an interrupt is generated if the respective conversion complete interrupt has been enabled, or, when SC1n[AIEN]=1.

The ADC module has the capability of automatically comparing the result of a conversion with the contents of the CV1 and CV2 registers. The compare function is enabled by setting SC2[ACFE] and operates in any of the conversion modes and configurations.

The ADC module has the capability of automatically averaging the result of multiple conversions. The hardware average function is enabled by setting SC3[AVGE] and operates in any of the conversion modes and configurations.

NOTE

For the chip specific modes of operation, see the power management information of this MCU.

24.4.1 Clock select and divide control

One of four clock sources can be selected as the clock source for the ADC module.

This clock source is then divided by a configurable value to generate the input clock ADCK, to the module. The clock is selected from one of the following sources by means of CFG1[ADICLK].

- Bus clock. This is the default selection following reset.
- Bus clock divided by two. For higher bus clock rates, this allows a maximum divide-by-16 of the bus clock using CFG1[ADIV].
- ALTCLK: As defined for this MCU. See the chip configuration information. Conversions are possible using ALTCLK as the input clock source while the MCU is in Normal Stop mode.
- Asynchronous clock (ADACK): This clock is generated from a clock source within the ADC module. When the ADACK clock source is selected, it is not required to be active prior to conversion start. When it is selected and it is not active prior to a conversion start CFG2[ADACKEN]=0, ADACK is activated at the start of a conversion and deactivated when conversions are terminated. In this case, there is an associated clock startup delay each time the clock source is re-activated. To avoid the conversion time variability and latency associated with the ADACK clock startup, set CFG2[ADACKEN]=1 and wait the worst-case startup time of 5 µs prior to initiating any conversions using the ADACK clock source. Conversions are possible using ADACK as the input clock source while the MCU is in Normal Stop mode. See Power Control for more information.

Whichever clock is selected, its frequency must fall within the specified frequency range for ADCK. If the available clocks are too slow, the ADC may not perform according to specifications. If the available clocks are too fast, the clock must be divided to the appropriate frequency. This divider is specified by CFG1[ADIV] and can be divide-by 1, 2, 4, or 8.

24.4.2 Voltage reference selection

The ADC can be configured to accept one of the two voltage reference pairs as the reference voltage (V_{REFSH} and V_{REFSL}) used for conversions.

Each pair contains a positive reference that must be between the minimum Ref Voltage High and V_{DDA}, and a ground reference that must be at the same potential as V_{SSA}. The two pairs are external (V_{REFH} and V_{REFL}) and alternate (V_{ALTH} and V_{ALTL}). These voltage references are selected using SC2[REFSEL]. The alternate (V_{ALTH} and V_{ALTL}) voltage reference pair may select additional external pins or internal sources depending on MCU configuration. See the chip configuration information on the voltage references specific to this MCU.

24.4.3 Hardware trigger and channel selects

The ADC module has a selectable asynchronous hardware conversion trigger, ADHWT, that is enabled when SC2[ADTRG] is set and a hardware trigger select event, ADHWTSn, has occurred.

This source is not available on all MCUs. See the Chip Configuration chapter for information on the ADHWT source and the ADHWTSn configurations specific to this MCU.

When an ADHWT source is available and hardware trigger is enabled, that is SC2[ADTRG]=1, a conversion is initiated on the rising-edge of ADHWT after a hardware trigger select event, that is, ADHWTSn, has occurred. If a conversion is in progress when a rising-edge of a trigger occurs, the rising-edge is ignored. In continuous convert configuration, only the initial rising-edge to launch continuous conversions is observed, and until conversion is aborted, the ADC continues to do conversions on the same SCn register that initiated the conversion. The hardware trigger function operates in conjunction with any of the conversion modes and configurations.

The hardware trigger select event, ADHWTSn, must be set prior to the receipt of the ADHWT signal. If these conditions are not met, the converter may ignore the trigger or use the incorrect configuration. If a hardware trigger select event is asserted during a conversion, it must stay asserted until the end of current conversion and remain set until the receipt of the ADHWT signal to trigger a new conversion. The channel and status fields selected for the conversion depend on the active trigger select signal:

- ADHWTSA active selects SC1A.
- ADHWTSn active selects SC1n.

Note

Asserting more than one hardware trigger select signal (ADHWTSn) at the same time results in unknown results. To avoid this, select only one hardware trigger select signal (ADHWTSn) prior to the next intended conversion.

When the conversion is completed, the result is placed in the Rn registers associated with the ADHWTSn received. For example:

- ADHWTSA active selects RA register
- ADHWTSn active selects Rn register

The conversion complete flag associated with the ADHWTSn received, that is, SC1n[COCO], is then set and an interrupt is generated if the respective conversion complete interrupt has been enabled, that is, SC1[AIEN]=1.

24.4.4 Conversion control

Conversions can be performed as determined by CFG1[MODE] as shown in the description of CFG1[MODE].

Conversions can be initiated by a software or hardware trigger.

In addition, the ADC module can be configured for:

- Low-power operation
- Long sample time
- Continuous conversion
- Hardware average
- Automatic compare of the conversion result to a software determined compare value

24.4.4.1 Initiating conversions

A conversion is initiated:

- Following a write to SC1A, with SC1n[ADCH] not all 1's, if software triggered operation is selected, that is, when SC2[ADTRG]=0.
- Following a hardware trigger, or ADHWT event, if hardware triggered operation is selected, that is, SC2[ADTRG]=1, and a hardware trigger select event, ADHWTSn, has occurred. The channel and status fields selected depend on the active trigger select signal:
  - ADHWTSA active selects SC1A.

- ADHWTSn active selects SC1n.
- if neither is active, the off condition is selected

Note

Selecting more than one ADHWTSn prior to a conversion completion will result in unknown results. To avoid this, select only one ADHWTSn prior to a conversion completion.

- Following the transfer of the result to the data registers when continuous conversion is enabled, that is, when SC3[ADCO] = 1.

If continuous conversions are enabled, a new conversion is automatically initiated after the completion of the current conversion. In software triggered operation, that is, when SC2[ADTRG] = 0, continuous conversions begin after SC1A is written and continue until aborted. In hardware triggered operation, that is, when SC2[ADTRG] = 1 and one ADHWTSn event has occurred, continuous conversions begin after a hardware trigger event and continue until aborted.

If hardware averaging is enabled, a new conversion is automatically initiated after the completion of the current conversion until the correct number of conversions are completed. In software triggered operation, conversions begin after SC1A is written. In hardware triggered operation, conversions begin after a hardware trigger. If continuous conversions are also enabled, a new set of conversions to be averaged are initiated following the last of the selected number of conversions.

24.4.4.2 Completing conversions

A conversion is completed when the result of the conversion is transferred into the data result registers, Rn. If the compare functions are disabled, this is indicated by setting of SC1n[COCO]. If hardware averaging is enabled, the respective SC1n[COCO] sets only if the last of the selected number of conversions is completed. If the compare function is enabled, the respective SC1n[COCO] sets and conversion result data is transferred only if the compare condition is true. If both hardware averaging and compare functions are enabled, then the respective SC1n[COCO] sets only if the last of the selected number of conversions is completed and the compare condition is true. An interrupt is generated if the respective SC1n[AIEN] is high at the time that the respective SC1n[COCO] is set.

24.4.4.3 Aborting conversions

Any conversion in progress is aborted when:

- Writing to SC1A while it is actively controlling a conversion, aborts the current conversion. In Software Trigger mode, when SC2[ADTRG]=0, a write to SC1A initiates a new conversion if SC1A[ADCH] is equal to a value other than all 1s. Writing to any of the SC1B–SC1n registers while that specific SC1B–SC1n register is actively controlling a conversion aborts the current conversion. The SC1(B-n) registers are not used for software trigger operation and therefore writes to the SC1(B-n) registers do not initiate a new conversion.
- A write to any ADC register besides the SC1A-SC1n registers occurs. This indicates that a change in mode of operation has occurred and the current conversion is therefore invalid.
- The MCU is reset or enters Low-Power Stop modes.
- The MCU enters Normal Stop mode with ADACK or Alternate Clock Sources not enabled.

When a conversion is aborted, the contents of the data registers, Rn, are not altered. The data registers continue to be the values transferred after the completion of the last successful conversion. If the conversion was aborted by a reset or Low-Power Stop modes, RA and Rn return to their reset states.

24.4.4.4 Power control

The ADC module remains in its idle state until a conversion is initiated. If ADACK is selected as the conversion clock source, but the asynchronous clock output is disabled, that is CFG2[ADACKEN]=0, the ADACK clock generator also remains in its idle state (disabled) until a conversion is initiated. If the asynchronous clock output is enabled, that is, CFG2[ADACKEN]=1, it remains active regardless of the state of the ADC or the MCU power mode.

Power consumption when the ADC is active can be reduced by setting CFG1[ADLPC]. This results in a lower maximum value for f_{ADCK}.

24.4.4.5 Sample time and total conversion time

For short sample, that is, when CFG1[ADLSMP]=0, there is a 2-cycle adder for first conversion over the base sample time of four ADCK cycles. For high-speed conversions, that is, when CFG2[ADHSC]=1, there is an additional 2-cycle adder on any conversion. The table below summarizes sample times for the possible ADC configurations.

ADC configuration			Sample time (ADCK cycles)	
CFG1[ADLSMP]	CFG2[ADLSTS]	CFG2[ADHSC]	First or Single	Subsequent
0	X	0	6	4
1	00	0	24	
1	01	0	16	
1	10	0	10	
1	11	0	6	
0	X	1	8	6
1	00	1	26	
1	01	1	18	
1	10	1	12	
1	11	1	8	

The total conversion time depends upon:

- The sample time as determined by CFG1[ADLSMP] and CFG2[ADLSTS]
- The MCU bus frequency
- The conversion mode, as determined by CFG1[MODE]
- The high-speed configuration, that is, CFG2[ADHSC]
- The frequency of the conversion clock, that is, f_{ADCK}.

CFG2[ADHSC] is used to configure a higher clock input frequency. This will allow faster overall conversion times. To meet internal ADC timing requirements, CFG2[ADHSC] adds additional ADCK cycles. Conversions with CFG2[ADHSC]=1 take two more ADCK cycles. CFG2[ADHSC] must be used when the ADCLK exceeds the limit for CFG2[ADHSC]=0.

After the module becomes active, sampling of the input begins.

1. CFG1[ADLSMP] and CFG2[ADLSTS] select between sample times based on the conversion mode that is selected.
2. When sampling is completed, the converter is isolated from the input channel and a successive approximation algorithm is applied to determine the digital value of the analog signal.
3. The result of the conversion is transferred to Rn upon completion of the conversion algorithm.

If the bus frequency is less than f_{ADCK}, precise sample time for continuous conversions cannot be guaranteed when short sample is enabled, that is, when CFG1[ADLSMP]=0.

The maximum total conversion time is determined by the clock source chosen and the divide ratio selected. The clock source is selectable by CFG1[ADICLK], and the divide ratio is specified by CFG1[ADIV].

The maximum total conversion time for all configurations is summarized in the equation below. See the following tables for the variables referenced in the equation.

$$\text{ConversionTime} = \text{SFCAdder} + \text{AverageNum} \times (\text{BCT} + \text{LSTAdder} + \text{HSCAdder})$$

Equation 1. Conversion time equation

Table 24-3. Single or first continuous time adder (SFCAdder)

CFG1[ADLSMP]	CFG2[ADACKEN]	CFG1[ADICLK]	Single or first continuous time adder (SFCAdder)
1	x	0x, 10	3 ADCK cycles + 5 bus clock cycles
1	1	11	3 ADCK cycles + 5 bus clock cycles[1]
1	0	11	5 µs + 3 ADCK cycles + 5 bus clock cycles
0	x	0x, 10	5 ADCK cycles + 5 bus clock cycles
0	1	11	5 ADCK cycles + 5 bus clock cycles[1]
0	0	11	5 µs + 5 ADCK cycles + 5 bus clock cycles

1. To achieve this time, CFG2[ADACKEN] must be 1 for at least 5 µs prior to the conversion is initiated.

Table 24-4. Average number factor (AverageNum)

SC3[AVGE]	SC3[AVGS]	Average number factor (AverageNum)
0	xx	1
1	00	4
1	01	8
1	10	16
1	11	32

Table 24-5. Base conversion time (BCT)

Mode	Base conversion time (BCT)
8b single-ended	17 ADCK cycles
10b single-ended	20 ADCK cycles
12b single-ended	20 ADCK cycles
16b single-ended	25 ADCK cycles

Table 24-6. Long sample time adder (LSTAdder)

CFG1[ADLSMP]	CFG2[ADLSTS]	Long sample time adder (LSTAdder)
0	xx	0 ADCK cycles
1	00	20 ADCK cycles
1	01	12 ADCK cycles
1	10	6 ADCK cycles
1	11	2 ADCK cycles

Table 24-7. High-speed conversion time adder (HSCAdder)

CFG2[ADHSC]	High-speed conversion time adder (HSCAdder)
0	0 ADCK cycles
1	2 ADCK cycles

Note

The ADCK frequency must be between f_{ADCK} minimum and f_{ADCK} maximum to meet ADC specifications.

24.4.4.6 Conversion time examples

The following examples use the Equation 1 on page 495, and the information provided in Table 24-3 through Table 24-7.

24.4.4.6.1 Typical conversion time configuration

A typical configuration for ADC conversion is:

- 10-bit mode, with the bus clock selected as the input clock source
- The input clock divide-by-1 ratio selected
- Bus frequency of 8 MHz
- Long sample time disabled
- High-speed conversion disabled

The conversion time for a single conversion is calculated by using the Equation 1 on page 495, and the information provided in Table 24-3 through Table 24-7. The table below lists the variables of Equation 1 on page 495.

Table 24-8. Typical conversion time

Variable	Time
SFCAdder	5 ADCK cycles + 5 bus clock cycles

Table continues on the next page...

Table 24-8. Typical conversion time (continued)

Variable	Time
AverageNum	1
BCT	20 ADCK cycles
LSTAdder	0
HSCAdder	0

The resulting conversion time is generated using the parameters listed in the preceding table. Therefore, for a bus clock and an ADCK frequency equal to 8 MHz, the resulting conversion time is 3.75 µs.

24.4.4.6.2 Long conversion time configuration

A configuration for long ADC conversion is:

- The input clock divide-by-8 ratio selected
- Bus frequency of 8 MHz
- Long sample time enabled
- Configured for longest adder
- High-speed conversion disabled
- Average enabled for 32 conversions

The conversion time for this conversion is calculated by using the Equation 1 on page 495, and the information provided in Table 24-3 through Table 24-7. The following table lists the variables of the Equation 1 on page 495.

Table 24-9. Typical conversion time

Variable	Time
SFCAdder	3 ADCK cycles + 5 bus clock cycles
AverageNum	32
BCT	34 ADCK cycles
LSTAdder	20 ADCK cycles
HSCAdder	0

The resulting conversion time is generated using the parameters listed in the preceding table. Therefore, for bus clock equal to 8 MHz and ADCK equal to 1 MHz, the resulting conversion time is 57.625 µs, that is, AverageNum. This results in a total conversion time of 1.844 ms.

24.4.4.6.3 Short conversion time configuration

A configuration for short ADC conversion is:

- 8-bit Single-Ended mode with the bus clock selected as the input clock source

- The input clock divide-by-1 ratio selected
- Bus frequency of 20 MHz
- Long sample time disabled
- High-speed conversion enabled

The conversion time for this conversion is calculated by using the Equation 1 on page 495, and the information provided in Table 24-3 through Table 24-7. The table below lists the variables of Equation 1 on page 495.

Table 24-10. Typical conversion time

Variable	Time
SFCAdder	5 ADCK cycles + 5 bus clock cycles
AverageNum	1
BCT	17 ADCK cycles
LSTAdder	0 ADCK cycles
HSCAdder	2

The resulting conversion time is generated using the parameters listed in in the preceding table. Therefore, for bus clock and ADCK frequency equal to 20 MHz, the resulting conversion time is 1.45 µs.

24.4.4.7 Hardware average function

The hardware average function can be enabled by setting SC3[AVGE]=1 to perform a hardware average of multiple conversions. The number of conversions is determined by the AVGS[1:0] bits, which can select 4, 8, 16, or 32 conversions to be averaged. While the hardware average function is in progress, SC2[ADACT] will be set.

After the selected input is sampled and converted, the result is placed in an accumulator from which an average is calculated once the selected number of conversions have been completed. When hardware averaging is selected, the completion of a single conversion will not set SC1n[COCO].

If the compare function is either disabled or evaluates true, after the selected number of conversions are completed, the average conversion result is transferred into the data result registers, Rn, and SC1n[COCO] is set. An ADC interrupt is generated upon the setting of SC1n[COCO] if the respective ADC interrupt is enabled, that is, SC1n[AIEN]=1.

Note

The hardware average function can perform conversions on a channel while the MCU is in Wait or Normal Stop modes. The ADC interrupt wakes the MCU when the hardware average is completed if SC1n[AIEN] is set.

24.4.5 Automatic compare function

The compare function can be configured to check whether the result is less than or greater-than-or-equal-to a single compare value, or, if the result falls within or outside a range determined by two compare values.

The compare mode is determined by SC2[ACFGT], SC2[ACREN], and the values in the compare value registers, CV1 and CV2. After the input is sampled and converted, the compare values in CV1 and CV2 are used as described in the following table. There are six Compare modes as shown in the following table.

Table 24-11. Compare modes

SC2[ACFGT]	SC2[ACREN]	ADCCV1 relative to ADCCV2	Function	Compare mode description
0	0	—	Less than threshold	Compare true if the result is less than the CV1 registers.
1	0	—	Greater than or equal to threshold	Compare true if the result is greater than or equal to CV1 registers.
0	1	Less than or equal	Outside range, not inclusive	Compare true if the result is less than CV1 **Or** the result is greater than CV2.
0	1	Greater than	Inside range, not inclusive	Compare true if the result is less than CV1 **And** the result is greater than CV2.
1	1	Less than or equal	Inside range, inclusive	Compare true if the result is greater than or equal to CV1 **And** the result is less than or equal to CV2.
1	1	Greater than	Outside range, inclusive	Compare true if the result is greater than or equal to CV1 **Or** the result is less than or equal to CV2.

With SC2[ACREN] =1, and if the value of CV1 is less than or equal to the value of CV2, then setting SC2[ACFGT] will select a trigger-if-inside-compare-range inclusive-of-endpoints function. Clearing SC2[ACFGT] will select a trigger-if-outside-compare-range, not-inclusive-of-endpoints function.

If CV1 is greater than CV2, setting SC2[ACFGT] will select a trigger-if-outside-compare-range, inclusive-of-endpoints function. Clearing SC2[ACFGT] will select a trigger-if-inside-compare-range, not-inclusive-of-endpoints function.

If the condition selected evaluates true, SC1n[COCO] is set.

Upon completion of a conversion while the compare function is enabled, if the compare condition is not true, SC1n[COCO] is not set and the conversion result data will not be transferred to the result register, Rn. If the hardware averaging function is enabled, the compare function compares the averaged result to the compare values. The same compare function definitions apply. An ADC interrupt is generated when SC1n[COCO] is set and the respective ADC interrupt is enabled, that is, SC1n[AIEN]=1.

Note

The compare function can monitor the voltage on a channel while the MCU is in Wait or Normal Stop modes. The ADC interrupt wakes the MCU when the compare condition is met.

24.4.6 Calibration function

>The ADC contains a self-calibration function that is required to achieve the specified accuracy.

Calibration must be run, or valid calibration values written, after any reset and before a conversion is initiated. The calibration function sets the offset calibration value and the plus-side calibration values. The offset calibration value is automatically stored in the ADC offset correction register (OFS), and the plus-side calibration values are automatically stored in the ADC plus-side calibration registers, CLPx. The user must configure the ADC correctly prior to calibration, and must generate the plus-side gain calibration results and store them in the ADC plus-side gain register (PG) after the calibration function completes.

Prior to calibration, the user must configure the ADC's clock source and frequency, low power configuration, voltage reference selection, sample time, and high speed configuration according to the application's clock source availability and needs. If the application uses the ADC in a wide variety of configurations, the configuration for which the highest accuracy is required should be selected, or multiple calibrations can be done for the different configurations. For best calibration results:

- Set hardware averaging to maximum, that is, SC3[AVGE]=1 and SC3[AVGS]=11 for an average of 32
- Set ADC clock frequency f_{ADCK} less than or equal to 4 MHz
- $V_{REFH}=V_{DDA}$
- Calibrate at nominal voltage and temperature

The input channel, conversion mode continuous function, compare function, resolution mode, and single-ended mode are all ignored during the calibration function.

To initiate calibration, the user sets SC3[CAL] and the calibration will automatically begin if the SC2[ADTRG] is 0. If SC2[ADTRG] is 1, SC3[CAL] will not get set and SC3[CALF] will be set. While calibration is active, no ADC register can be written and no stop mode may be entered, or the calibration routine will be aborted causing SC3[CAL] to clear and SC3[CALF] to set. At the end of a calibration sequence, SC1n[COCO] will be set. SC1n[AIEN] can be used to allow an interrupt to occur at the end of a calibration sequence. At the end of the calibration routine, if SC3[CALF] is not set, the automatic calibration routine is completed successfully.

To complete calibration, the user must generate the gain calibration values using the following procedure:

1. Initialize or clear a 16-bit variable in RAM.
2. Add the plus-side calibration results CLP0, CLP1, CLP2, CLP3, CLP4, and CLPS to the variable.
3. Divide the variable by two.
4. Set the MSB of the variable.
5. The previous two steps can be achieved by setting the carry bit, rotating to the right through the carry bit on the high byte and again on the low byte.
6. Store the value in the plus-side gain calibration register PG.

When calibration is complete, the user may reconfigure and use the ADC as desired. A second calibration may also be performed, if desired, by clearing and again setting SC3[CAL].

Overall, the calibration routine may take as many as 14k ADCK cycles and 100 bus cycles, depending on the results and the clock source chosen. For an 8 MHz clock source, this length amounts to about 1.7 ms. To reduce this latency, the calibration values, which are offset, plus-side gain, and plus-side calibration values, may be stored in flash memory after an initial calibration and recovered prior to the first ADC conversion. This method can reduce the calibration latency to 20 register store operations on all subsequent power, reset, or Low-Power Stop mode recoveries.

Further information on the calibration procedure can be found in the Calibration section of AN3949: ADC16 Calibration Procedure and Programmable Delay Block Synchronization.

24.4.7 User-defined offset function

OFS contains the user-selected or calibration-generated offset error correction value.

This register is a 2’s complement, left-justified. The value in OFS is subtracted from the conversion and the result is transferred into the result registers, Rn. If the result is greater than the maximum or less than the minimum result value, it is forced to the appropriate limit for the current mode of operation.

The formatting of the OFS is different from the data result register, Rn, to preserve the resolution of the calibration value regardless of the conversion mode selected. Lower order bits are ignored in lower resolution modes. For example, in 8-bit single-ended mode, OFS[14:7] are subtracted from D[7:0]; OFS[15] indicates the sign (negative numbers are effectively added to the result) and OFS[6:0] are ignored. In 16-bit single-ended mode, there is no field in the OFS corresponding to the least significant result D[0], so odd values, such as -1 or +1, cannot be subtracted from the result.

OFS is automatically set according to calibration requirements once the self-calibration sequence is done, that is, SC3[CAL] is cleared. The user may write to OFS to override the calibration result if desired. If the OFS is written by the user to a value that is different from the calibration value, the ADC error specifications may not be met. Storing the value generated by the calibration function in memory before overwriting with a user-specified value is recommended.

Note

There is an effective limit to the values of offset that can be set by the user. If the magnitude of the offset is too high, the results of the conversions will cap off at the limits.

The offset calibration function may be employed by the user to remove application offsets or DC bias values. OFS may be written with a number in 2's complement format and this offset will be subtracted from the result, or hardware averaged value. To add an offset, store the negative offset in 2's complement format and the effect will be an addition. An offset correction that results in an out-of-range value will be forced to the minimum or maximum value. The minimum value for single-ended conversions is 0x0000.

To preserve accuracy, the calibrated offset value initially stored in OFS must be added to the user-defined offset. For applications that may change the offset repeatedly during operation, store the initial offset calibration value in flash so it can be recovered and added to any user offset adjustment value and the sum stored in OFS.

24.4.8 Temperature sensor

The ADC module includes a temperature sensor whose output is connected to one of the ADC analog channel inputs.

The following equation provides an approximate transfer function of the temperature sensor.

$$Temp = 25 - \left(\left(V_{TEMP} - V_{TEMP25} \right) \div m \right)$$

Equation 2. Approximate transfer function of the temperature sensor

where:

- V_{TEMP} is the voltage of the temperature sensor channel at the ambient temperature.
- V_{TEMP25} is the voltage of the temperature sensor channel at 25 °C.
- m is referred as temperature sensor slope in the device data sheet. It is the hot or cold voltage versus temperature slope in V/°C.

For temperature calculations, use the V_{TEMP25} and temperature sensor slope values from the ADC Electricals table.

In application code, the user reads the temperature sensor channel, calculates V_{TEMP}, and compares to V_{TEMP25}. If V_{TEMP} is greater than V_{TEMP25} the cold slope value is applied in the preceding equation. If V_{TEMP} is less than V_{TEMP25}, the hot slope value is applied in the preceding equation. ADC Electricals table may only specify one temperature sensor slope value. In that case, the user could use the same slope for the calculation across the operational temperature range.

For more information on using the temperature sensor, see the application note titled *Temperature Sensor for the HCS08 Microcontroller Family* (document AN3031).

24.4.9 MCU wait mode operation

Wait mode is a lower-power consumption Standby mode from which recovery is fast because the clock sources remain active.

If a conversion is in progress when the MCU enters Wait mode, it continues until completion. Conversions can be initiated while the MCU is in Wait mode by means of the hardware trigger or if continuous conversions are enabled.

The bus clock, ADACK are available as conversion clock sources while in Wait mode. The use of ALTCLK as the conversion clock source in Wait is dependent on the definition of ALTCLK for this MCU. See the Chip Configuration information on ALTCLK specific to this MCU.

If the compare and hardware averaging functions are disabled, a conversion complete event sets SC1n[COCO] and generates an ADC interrupt to wake the MCU from Wait mode if the respective ADC interrupt is enabled, that is, when SC1n[AIEN]=1. If the hardware averaging function is enabled, SC1n[COCO] will set, and generate an interrupt if enabled, when the selected number of conversions are completed. If the compare function is enabled, SC1n[COCO] will set, and generate an interrupt if enabled, only if the compare conditions are met. If a single conversion is selected and the compare trigger is not met, the ADC will return to its idle state and cannot wake the MCU from Wait mode unless a new conversion is initiated by the hardware trigger.

24.4.10 MCU Normal Stop mode operation

Stop mode is a low-power consumption Standby mode during which most or all clock sources on the MCU are disabled.

24.4.10.1 Normal Stop mode with

If the selected as the conversion clock, executing a stop instruction aborts the current conversion and places the ADC in its Idle state. The contents of the ADC registers, including Rn, are unaffected by Normal Stop mode. After exiting from Normal Stop mode, a software or hardware trigger is required to resume conversions.

24.4.10.2 Normal Stop mode with ADACK enabled

If ADACK selected as the conversion clock, the ADC continues operation during Normal Stop mode. See the chip configuration chapter for configuration information for this MCU.

If a conversion is in progress when the MCU enters Normal Stop mode, it continues until completion. Conversions can be initiated while the MCU is in Normal Stop mode by means of the hardware trigger or if continuous conversions are enabled.

If the compare and hardware averaging functions are disabled, a conversion complete event sets SC1n[COCO] and generates an ADC interrupt to wake the MCU from Normal Stop mode if the respective ADC interrupt is enabled, that is, when SC1n[AIEN]=1. The result register, Rn, will contain the data from the first completed conversion that occurred during Normal Stop mode. If the hardware averaging function is enabled, SC1n[COCO] will set, and generate an interrupt if enabled, when the selected number of conversions are completed. If the compare function is enabled, SC1n[COCO] will set, and generate an interrupt if enabled, only if the compare conditions are met. If a single conversion is

selected and the compare is not true, the ADC will return to its Idle state and cannot wake the MCU from Normal Stop mode unless a new conversion is initiated by another hardware trigger.

24.4.11 MCU Low-Power Stop mode operation

The ADC module is automatically disabled when the MCU enters Low-Power Stop mode.

All module registers contain their reset values following exit from Low-Power Stop mode. Therefore, the module must be re-enabled and re-configured following exit from Low-Power Stop mode.

NOTE

For the chip specific modes of operation, see the power management information for the device.

24.5 Initialization information

This section gives an example that provides some basic direction on how to initialize and configure the ADC module.

The user can configure the module for 16-bit, 12-bit, 10-bit, or 8-bit single-ended resolution, single or continuous conversion, and a polled or interrupt approach, among many other options. For information used in this example, refer to Table 24-6, Table 24-7, and Table 24-8.

Note

Hexadecimal values are designated by a preceding 0x, binary values designated by a preceding %, and decimal values have no preceding character.

24.5.1 ADC module initialization example

24.5.1.1 Initialization sequence

Before the ADC module can be used to complete conversions, an initialization procedure must be performed. A typical sequence is:

1. Calibrate the ADC by following the calibration instructions in Calibration function.
2. Update CFG to select the input clock source and the divide ratio used to generate ADCK. This register is also used for selecting sample time and low-power configuration.
3. Update SC2 to select the conversion trigger, hardware or software, and compare function options, if enabled.
4. Update SC3 to select whether conversions will be continuous or completed only once (ADCO) and whether to perform hardware averaging.
5. Update SC1:SC1n registers to enable or disable conversion complete interrupts. Also, select the input channel which can be used to perform conversions.

24.5.1.2 Pseudo-code example

In this example, the ADC module is set up with interrupts enabled to perform a single 10-bit conversion at low-power with a long sample time on input channel 1, where ADCK is derived from the bus clock divided by 1.

CFG1 = 0x98 (%10011000)

```
        Bit 7   ADLPC   1       Configures for low power, lowers maximum clock speed.
        Bit 6:5 ADIV    00      Sets the ADCK to the input clock ÷ 1.
        Bit 4   ADLSMP  1       Configures for long sample time.
        Bit 3:2   MODE   10     Selects the single-ended 10-bit conversion.
        Bit 1:0   ADICLK   00     Selects the bus clock.
```

SC2 = 0x00 (%00000000)

```
        Bit 7   ADACT   0        Flag indicates if a conversion is in progress.
        Bit 6   ADTRG   0        Software trigger selected.
        Bit 5   ACFE    0        Compare function disabled.
        Bit 4   ACFGT   0        Not used in this example.
        Bit 3   ACREN   0       Compare range disabled.
 Bit 2 DMAEN 0 DMA request disabled.
        Bit 1:0 REFSEL  00      Selects default voltage reference pin pair (External pins
VREFH and VREFL).
```

SC1A = 0x41 (%01000001)

```
        Bit 7   COCO    0       Read-only flag which is set when a conversion completes.
        Bit 6   AIEN    1       Conversion complete interrupt enabled.

        Bit 4:0 ADCH    00001   Input channel 1 selected as ADC input channel.
```

RA = 0xxx

```
        Holds results of conversion.
```

CV = 0xxx

```
Holds compare value when compare function enabled.
```

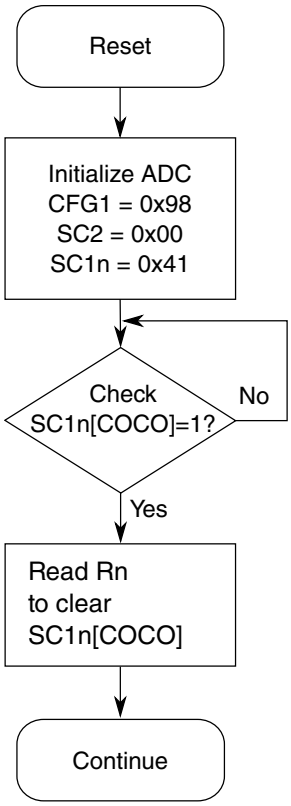


Figure 24-2. Initialization flowchart example

24.6 Application information

The ADC has been designed to be integrated into a microcontroller for use in embedded control applications requiring an ADC.

For guidance on selecting optimum external component values and converter parameters see AN4373: Cookbook for SAR ADC Measurements.

24.6.1 External pins and routing

24.6.1.1 Analog supply pins

Depending on the device, the analog power and ground supplies, V_{DDA} and V_{SSA}, of the ADC module are available as:

- V_{DDA} and V_{SSA} available as separate pins—When available on a separate pin, both V_{DDA} and V_{SSA} must be connected to the same voltage potential as their corresponding MCU digital supply, V_{DD} and V_{SS}, and must be routed carefully for maximum noise immunity and bypass capacitors placed as near as possible to the package.

- V_{SSA} is shared on the same pin as the MCU digital V_{SS}.
- V_{SSA} and V_{DDA} are shared with the MCU digital supply pins—In these cases, there are separate pads for the analog supplies bonded to the same pin as the corresponding digital supply so that some degree of isolation between the supplies is maintained.

If separate power supplies are used for analog and digital power, the ground connection between these supplies must be at the V_{SSA} pin. This must be the only ground connection between these supplies, if possible. V_{SSA} makes a good single point ground location.

24.6.1.2 Analog voltage reference pins

In addition to the analog supplies, the ADC module has connections for two reference voltage inputs used by the converter:

- V_{REFSH} is the high reference voltage for the converter.
- V_{REFSL} is the low reference voltage for the converter.

The ADC can be configured to accept one of two voltage reference pairs for V_{REFSH} and V_{REFSL}. Each pair contains a positive reference and a ground reference. The two pairs are external, V_{REFH} and V_{REFL} and alternate, V_{ALTH} and V_{ALTL}. These voltage references are selected using SC2[REFSEL]. The alternate voltage reference pair, V_{ALTH} and V_{ALTL}, may select additional external pins or internal sources based on MCU configuration. See the chip configuration information on the voltage references specific to this MCU.

In some packages, the external or alternate pairs are connected in the package to V_{DDA} and V_{SSA}, respectively. One of these positive references may be shared on the same pin as V_{DDA} on some devices. One of these ground references may be shared on the same pin as V_{SSA} on some devices.

If externally available, the positive reference may be connected to the same potential as V_{DDA} or may be driven by an external source to a level between the minimum Ref Voltage High and the V_{DDA} potential. The positive reference must never exceed V_{DDA}. If externally available, the ground reference must be connected to the same voltage potential as V_{SSA}. The voltage reference pairs must be routed carefully for maximum noise immunity and bypass capacitors placed as near as possible to the package.

AC current in the form of current spikes required to supply charge to the capacitor array at each successive approximation step is drawn through the V_{REFH} and V_{REFL} loop. The best external component to meet this current demand is a 0.1 µF capacitor with good

high-frequency characteristics. This capacitor is connected between V_{REFH} and V_{REFL} and must be placed as near as possible to the package pins. Resistance in the path is not recommended because the current causes a voltage drop that could result in conversion errors. Inductance in this path must be minimum, that is, parasitic only.

24.6.1.3 Analog input pins

The external analog inputs are typically shared with digital I/O pins on MCU devices.

Empirical data shows that capacitors on the analog inputs improve performance in the presence of noise or when the source impedance is high. Use of 0.01 μF capacitors with good high-frequency characteristics is sufficient. These capacitors are not necessary in all cases, but when used, they must be placed as near as possible to the package pins and be referenced to V_{SSA}.

For proper conversion, the input voltage must fall between V_{REFH} and V_{REFL}. If the input is equal to or exceeds V_{REFH}, the converter circuit converts the signal to 0xFFF, which is full scale 12-bit representation, 0x3FF, which is full scale 10-bit representation, or 0xFF, which is full scale 8-bit representation. If the input is equal to or less than V_{REFL}, the converter circuit converts it to 0x000. Input voltages between V_{REFH} and V_{REFL} are straight-line linear conversions. There is a brief current associated with V_{REFL} when the sampling capacitor is charging.

For minimal loss of accuracy due to current injection, pins adjacent to the analog input pins must not be transitioning during conversions.

24.6.2 Sources of error

24.6.2.1 Sampling error

For proper conversions, the input must be sampled long enough to achieve the proper accuracy.

$$RAS + RADIN = SC / (FMAX * NUMTAU * CADIN)$$

Figure 24-3. Sampling equation

Where:

RAS = External analog source resistance

SC = Number of ADCK cycles used during sample window

CADIN = Internal ADC input capacitance

NUMTAU = -ln(LSBERR / 2^N)

LSBERR = value of acceptable sampling error in LSBs

N = 8 in 8-bit mode, 10 in 10-bit mode, 12 in 12-bit mode or 16 in 16-bit mode

Higher source resistances or higher-accuracy sampling is possible by setting CFG1[ADLSMP] and changing CFG2[ADLSTS] to increase the sample window, or decreasing ADCK frequency to increase sample time.

24.6.2.2 Pin leakage error

Leakage on the I/O pins can cause conversion error if the external analog source resistance, R_{AS}, is high. If this error cannot be tolerated by the application, keep R_{AS} lower than V_{REFH} / (4 × I_{LEAK} × 2^N) for less than 1/4 LSB leakage error, where N = 8 in 8-bit mode, 10 in 10-bit mode, 12 in 12-bit mode, or 16 in 16-bit mode.

24.6.2.3 Noise-induced errors

System noise that occurs during the sample or conversion process can affect the accuracy of the conversion. The ADC accuracy numbers are guaranteed as specified only if the following conditions are met:

- There is a 0.1 μF low-ESR capacitor from V_{REFH} to V_{REFL}.
- There is a 0.1 μF low-ESR capacitor from V_{DDA} to V_{SSA}.
- If inductive isolation is used from the primary supply, an additional 1 μF capacitor is placed from V_{DDA} to V_{SSA}.
- V_{SSA}, and V_{REFL}, if connected, is connected to V_{SS} at a quiet point in the ground plane.
- Operate the MCU in Wait or Normal Stop mode before initiating (hardware-triggered conversions) or immediately after initiating (hardware- or software-triggered conversions) the ADC conversion.

- For software triggered conversions, immediately follow the write to SC1 with a Wait instruction or Stop instruction.
- For Normal Stop mode operation, select ADACK as the clock source. Operation in Normal Stop reduces V_{DD} noise but increases effective conversion time due to stop recovery.

- There is no I/O switching, input or output, on the MCU during the conversion.

There are some situations where external system activity causes radiated or conducted noise emissions or excessive V_{DD} noise is coupled into the ADC. In these situations, or when the MCU cannot be placed in Wait or Normal Stop mode, or I/O activity cannot be halted, the following actions may reduce the effect of noise on the accuracy:

- Place a 0.01 µF capacitor (C_{AS}) on the selected input channel to V_{REFL} or V_{SSA}. This improves noise issues, but affects the sample rate based on the external analog source resistance.
- Average the result by converting the analog input many times in succession and dividing the sum of the results. Four samples are required to eliminate the effect of a 1 LSB, one-time error.
- Reduce the effect of synchronous noise by operating off the asynchronous clock, that is, ADACK, and averaging. Noise that is synchronous to ADCK cannot be averaged out.

24.6.2.4 Code width and quantization error

The ADC quantizes the ideal straight-line transfer function into 65536 steps in the 16-bit mode). Each step ideally has the same height, that is, 1 code, and width. The width is defined as the delta between the transition points to one code and the next. The ideal code width for an N-bit converter, where N can be 16, 12, 10, or 8, defined as 1 LSB, is:

$$1LSB = \left(V_{REFH} \right) / 2^N$$

Equation 3. Ideal code width for an N-bit converter

There is an inherent quantization error due to the digitization of the result. For 8-bit, 10-bit, or 12-bit conversions, the code transitions when the voltage is at the midpoint between the points where the straight line transfer function is exactly represented by the actual transfer function. Therefore, the quantization error will be ± 1/2 LSB in 8-bit, 10-bit, or 12-bit modes. As a consequence, however, the code width of the first (0x000) conversion is only 1/2 LSB and the code width of the last (0xFF or 0x3FF) is 1.5 LSB.

For 16-bit conversions, the code transitions only after the full code width is present, so the quantization error is -1 LSB to 0 LSB and the code width of each step is 1 LSB.

24.6.2.5 Linearity errors

The ADC may also exhibit non-linearity of several forms. Every effort has been made to reduce these errors, but the system designers must be aware of these errors because they affect overall accuracy:

- Zero-scale error (E_{ZS}), sometimes called offset: This error is defined as the difference between the actual code width of the first conversion and the ideal code width. This is 1/2 LSB in 8-bit, 10-bit, or 12-bit modes and 1 LSB in 16-bit mode. If the first conversion is 0x001, the difference between the actual 0x001 code width and its ideal (1 LSB) is used.
- Full-scale error (E_{FS}): This error is defined as the difference between the actual code width of the last conversion and the ideal code width. This is 1.5 LSB in 8-bit, 10-bit, or 12-bit modes and 1 LSB in 16-bit mode. If the last conversion is 0x3FE, the difference between the actual 0x3FE code width and its ideal (1 LSB) is used.
- Differential non-linearity (DNL): This error is defined as the worst-case difference between the actual code width and the ideal code width for all conversions.
- Integral non-linearity (INL): This error is defined as the highest-value or absolute value that the running sum of DNL achieves. More simply, this is the worst-case difference of the actual transition voltage to a given code and its corresponding ideal transition voltage, for all codes.
- Total unadjusted error (TUE): This error is defined as the difference between the actual transfer function and the ideal straight-line transfer function and includes all forms of error.

24.6.2.6 Code jitter, non-monotonicity, and missing codes

Analog-to-digital converters are susceptible to three special forms of error:

- Code jitter: Code jitter is when, at certain points, a given input voltage converts to one of the two values when sampled repeatedly. Ideally, when the input voltage is infinitesimally smaller than the transition voltage, the converter yields the lower code, and vice-versa. However, even small amounts of system noise can cause the converter to be indeterminate, between two codes, for a range of input voltages around the transition voltage.

This error may be reduced by repeatedly sampling the input and averaging the result. Additionally, the techniques discussed in Noise-induced errors reduces this error.

- Non-monotonicity: Non-monotonicity is defined as when, except for code jitter, the converter converts to a lower code for a higher input voltage.
- Missing codes: Missing codes are those values never converted for any input value.

In 8-bit or 10-bit mode, the ADC is guaranteed to be monotonic and have no missing codes.

Chapter 25
12-bit Cyclic Analog-to-Digital Converter (ADC12)

25.1 Introduction

25.1.1 Overview

The analog-to-digital converter (ADC) is one dual 12-bit ADC in which each ADC converter has a separate voltage reference and control block.

25.1.2 Features

The analog-to-digital (ADC) converter function consists of two separate analog-to-digital converters, each with eight analog inputs and its own sample and hold circuit. A common digital control module configures and controls the functioning of the converters. ADC features include:

- 12-bit resolution
- Designed for maximum ADC clock frequency of 20 MHz with 50 ns period
- Sampling rate up to 6.67 million samples per second[1]
- Single conversion time of 8.5 ADC clock cycles (8.5 × 50 ns = 425ns)
- Additional conversion time of 6 ADC clock cycles (6 × 50 ns = 300ns)
- Eight conversions in 26.5 ADC clock cycles (26.5 × 50 ns = 1.325µs) using parallel mode
- Can be synchronized to other peripherals that are connected to an internal Inter-Peripheral Crossbar module, such as the PWM, through the SYNC0/1 input signal

1. In loop mode, the time between each conversion is 6 ADC clock cycles (300ns). Using simultaneous conversion, two samples can be obtained in 300ns. Samples per second is calculated according to 300ns per two samples or 6,666,667 samples per second.

- Sequentially scans and stores up to sixteen measurements
- Scans and stores up to eight measurements each on two ADC converters operating simultaneously and in parallel
- Scans and stores up to eight measurements each on two ADC converters operating asynchronously to each other in parallel
- A scan can pause and await new SYNC input prior to continuing
- Gains the input signal by x1, x2, or x4
- Optional interrupts at end of scan if an out-of-range limit is exceeded or there is a zero crossing
- Optional DMA function to transfer conversion data at the end of a scan or when a sample is ready to be read.
- Optional sample correction by subtracting a pre-programmed offset value
- Signed or unsigned result
- Single-ended or differential inputs
- PWM outputs with hysteresis for three of the analog inputs

25.1.3 Block Diagram

The following figure illustrates the dual ADC configuration.

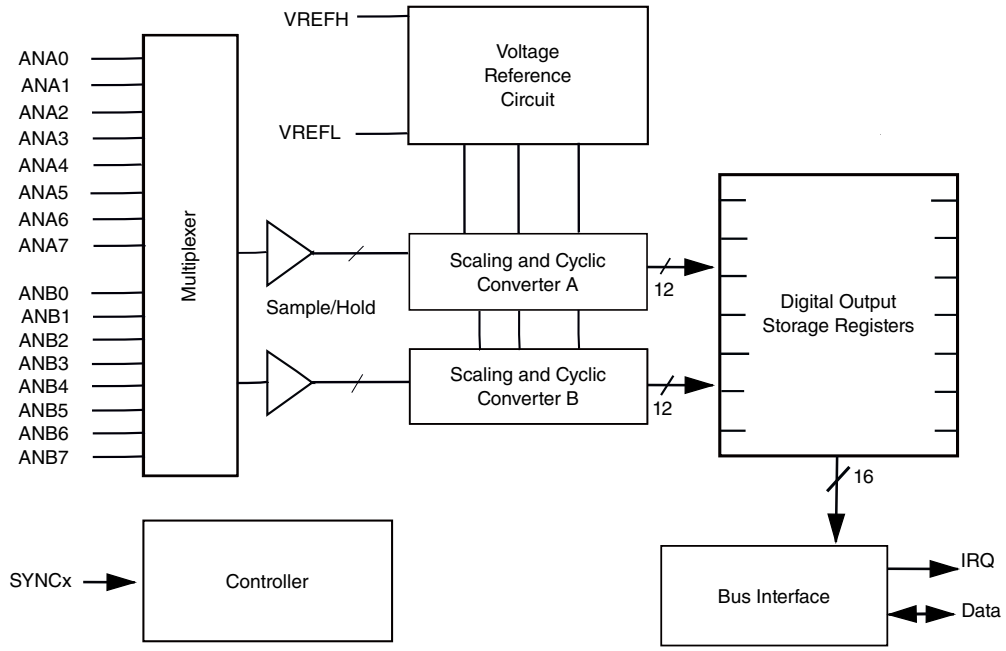


Figure 25-1. Dual ADC Block Diagram

25.2 Signal Descriptions

25.2.1 Overview

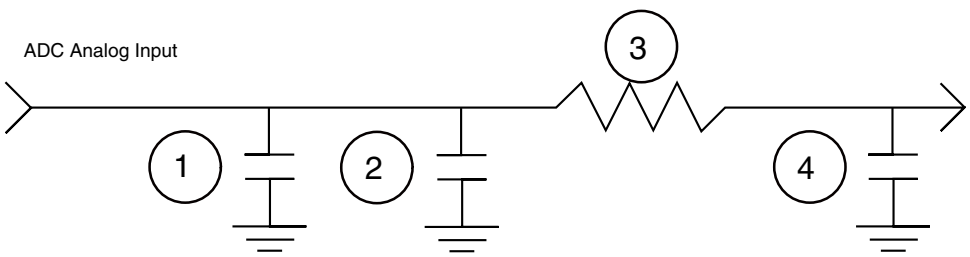
Table 25-1. External Signal Properties

Name	I/O Type	Function	Reset State	Notes
ANA0–ANB7	I	Analog Input Pins	n/a	—
VREFH	I	Voltage Reference Pin	n/a	Selectable between VDDA and ANA2
VREFL	I	Voltage Reference Pin	n/a	Selectable between VSSA and ANA3
VREFH	I	Voltage Reference Pin	n/a	Selectable between VDDA and ANB2
VREFL	I	Voltage Reference Pin	n/a	Selectable between VSSA and ANB3
VDDA	Supply	ADC Power	n/a	—
VSSA	Supply	ADC Ground	n/a	—

25.2.2 External Signal Descriptions

25.2.2.1 Analog Input Pins (ANA[0:7] and ANB[0:7])

Each ADC module has sixteen analog input pins that are subdivided into two sets of eight (ANA[0:7] and ANB[0:7]), each with their own S/H unit and converter. This configuration allows simultaneous sampling of two selected channels, one from each subgroup. Sequential scans have access to all sixteen analog inputs. During parallel scans, each ADC converter has access to its eight analog inputs. An equivalent circuit for an analog input is shown below:



1. Parasitic capacitance due to package, pin-to-pin, and pin-to-package base coupling. 1.8pf
2. Parasitic capacitance due to the chip bond pad, ESD protection devices, and signal routing. 2.04pf
3. Equivalent resistance for the ESD isolation resistor and the channel select multiplexer. 500Ω
4. Sampling capacitor at the sample and hold circuit. 1pf

Figure 25-2. Equivalent Analog Input Circuit

25.2.2.2 Voltage Reference Pins (VREFH and VREFL)

The voltage difference between V_{REFH} and V_{REFL} provides the reference voltage against which all analog inputs are measured. V_{REFH} is nominally set to V_{DDA}. V_{REFL} is nominally set to 0V. Any external reference voltage should come from a low noise filtered source. The external reference source should provide up to 1mA of reference current. illustrates the internal workings of the ADC voltage reference circuit. V_{REFH} must be noise filtered. A minimum configuration is shown in the figure.

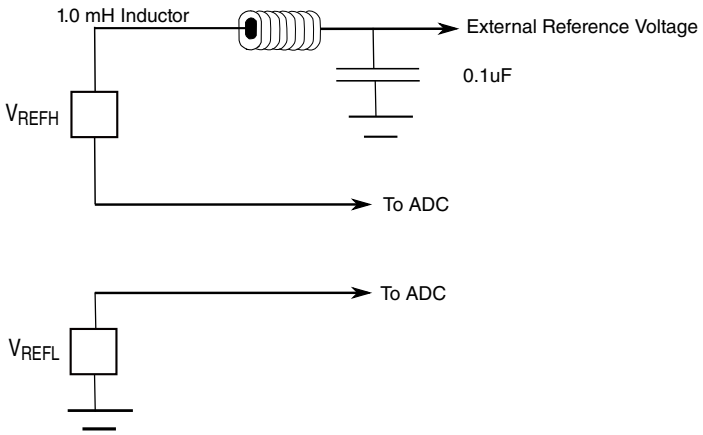


Figure 25-3. ADC Voltage Reference Circuit

When V_{DDA} is used as V_{REFH}, measurements are made with respect to the amplitude of V_{DDA}. Special precautions must be taken to assure that the voltage applied to V_{REFH} is as noise free as possible. Any noise residing on the V_{REFH} voltage is directly transferred to the digital result.

Dedicated power supply pins, V_{DDA} and V_{SSA}, are provided to reduce noise coupling and to improve accuracy. The power to these pins should come from a low noise filtered source. Uncoupling capacitors should be connected between V_{DDA} and V_{SSA}.

V_{SS} is shared between both analog and digital circuitry.

25.3 Memory Map and Registers

ADC memory map

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
E500	ADC Control Register 1 (ADC12_CTRL1)	16	R/W	5005h	25.3.1/523
E501	ADC Control Register 2 (ADC12_CTRL2)	16	R/W	5044h	25.3.2/527
E502	ADC Zero Crossing Control 1 Register (ADC12_ZXCTRL1)	16	R/W	0000h	25.3.3/529
E503	ADC Zero Crossing Control 2 Register (ADC12_ZXCTRL2)	16	R/W	0000h	25.3.4/531
E504	ADC Channel List Register 1 (ADC12_CLIST1)	16	R/W	3210h	25.3.5/532
E505	ADC Channel List Register 2 (ADC12_CLIST2)	16	R/W	7654h	25.3.6/534
E506	ADC Channel List Register 3 (ADC12_CLIST3)	16	R/W	BA98h	25.3.7/536
E507	ADC Channel List Register 4 (ADC12_CLIST4)	16	R/W	FEDCh	25.3.8/537
E508	ADC Sample Disable Register (ADC12_SDIS)	16	R/W	F0F0h	25.3.9/539
E509	ADC Status Register (ADC12_STAT)	16	R	0000h	25.3.10/ 540

Table continues on the next page...

ADC memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
E50A	ADC Ready Register (ADC12_RDY)	16	R	0000h	25.3.11/ 542
E50B	ADC Low Limit Status Register (ADC12_LOLIMSTAT)	16	w1c	0000h	25.3.12/ 542
E50C	ADC High Limit Status Register (ADC12_HILIMSTAT)	16	w1c	0000h	25.3.13/ 543
E50D	ADC Zero Crossing Status Register (ADC12_ZXSTAT)	16	w1c	0000h	25.3.14/ 543
E50E	ADC Result Registers with sign extension (ADC12_RSLT0)	16	R/W	0000h	25.3.15/ 544
E50F	ADC Result Registers with sign extension (ADC12_RSLT1)	16	R/W	0000h	25.3.15/ 544
E510	ADC Result Registers with sign extension (ADC12_RSLT2)	16	R/W	0000h	25.3.15/ 544
E511	ADC Result Registers with sign extension (ADC12_RSLT3)	16	R/W	0000h	25.3.15/ 544
E512	ADC Result Registers with sign extension (ADC12_RSLT4)	16	R/W	0000h	25.3.15/ 544
E513	ADC Result Registers with sign extension (ADC12_RSLT5)	16	R/W	0000h	25.3.15/ 544
E514	ADC Result Registers with sign extension (ADC12_RSLT6)	16	R/W	0000h	25.3.15/ 544
E515	ADC Result Registers with sign extension (ADC12_RSLT7)	16	R/W	0000h	25.3.15/ 544
E516	ADC Result Registers with sign extension (ADC12_RSLT8)	16	R/W	0000h	25.3.15/ 544
E517	ADC Result Registers with sign extension (ADC12_RSLT9)	16	R/W	0000h	25.3.15/ 544
E518	ADC Result Registers with sign extension (ADC12_RSLT10)	16	R/W	0000h	25.3.15/ 544
E519	ADC Result Registers with sign extension (ADC12_RSLT11)	16	R/W	0000h	25.3.15/ 544
E51A	ADC Result Registers with sign extension (ADC12_RSLT12)	16	R/W	0000h	25.3.15/ 544
E51B	ADC Result Registers with sign extension (ADC12_RSLT13)	16	R/W	0000h	25.3.15/ 544
E51C	ADC Result Registers with sign extension (ADC12_RSLT14)	16	R/W	0000h	25.3.15/ 544
E51D	ADC Result Registers with sign extension (ADC12_RSLT15)	16	R/W	0000h	25.3.15/ 544
E51E	ADC Low Limit Registers (ADC12_LOLIM0)	16	R/W	0000h	25.3.16/ 545
E51F	ADC Low Limit Registers (ADC12_LOLIM1)	16	R/W	0000h	25.3.16/ 545

Table continues on the next page...

ADC memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
E520	ADC Low Limit Registers (ADC12_LOLIM2)	16	R/W	0000h	25.3.16/ 545
E521	ADC Low Limit Registers (ADC12_LOLIM3)	16	R/W	0000h	25.3.16/ 545
E522	ADC Low Limit Registers (ADC12_LOLIM4)	16	R/W	0000h	25.3.16/ 545
E523	ADC Low Limit Registers (ADC12_LOLIM5)	16	R/W	0000h	25.3.16/ 545
E524	ADC Low Limit Registers (ADC12_LOLIM6)	16	R/W	0000h	25.3.16/ 545
E525	ADC Low Limit Registers (ADC12_LOLIM7)	16	R/W	0000h	25.3.16/ 545
E526	ADC Low Limit Registers (ADC12_LOLIM8)	16	R/W	0000h	25.3.16/ 545
E527	ADC Low Limit Registers (ADC12_LOLIM9)	16	R/W	0000h	25.3.16/ 545
E528	ADC Low Limit Registers (ADC12_LOLIM10)	16	R/W	0000h	25.3.16/ 545
E529	ADC Low Limit Registers (ADC12_LOLIM11)	16	R/W	0000h	25.3.16/ 545
E52A	ADC Low Limit Registers (ADC12_LOLIM12)	16	R/W	0000h	25.3.16/ 545
E52B	ADC Low Limit Registers (ADC12_LOLIM13)	16	R/W	0000h	25.3.16/ 545
E52C	ADC Low Limit Registers (ADC12_LOLIM14)	16	R/W	0000h	25.3.16/ 545
E52D	ADC Low Limit Registers (ADC12_LOLIM15)	16	R/W	0000h	25.3.16/ 545
E52E	ADC High Limit Registers (ADC12_HILIM0)	16	R/W	7FF8h	25.3.17/ 546
E52F	ADC High Limit Registers (ADC12_HILIM1)	16	R/W	7FF8h	25.3.17/ 546
E530	ADC High Limit Registers (ADC12_HILIM2)	16	R/W	7FF8h	25.3.17/ 546
E531	ADC High Limit Registers (ADC12_HILIM3)	16	R/W	7FF8h	25.3.17/ 546
E532	ADC High Limit Registers (ADC12_HILIM4)	16	R/W	7FF8h	25.3.17/ 546
E533	ADC High Limit Registers (ADC12_HILIM5)	16	R/W	7FF8h	25.3.17/ 546
E534	ADC High Limit Registers (ADC12_HILIM6)	16	R/W	7FF8h	25.3.17/ 546
E535	ADC High Limit Registers (ADC12_HILIM7)	16	R/W	7FF8h	25.3.17/ 546

Table continues on the next page...

ADC memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
E536	ADC High Limit Registers (ADC12_HILIM8)	16	R/W	7FF8h	25.3.17/ 546
E537	ADC High Limit Registers (ADC12_HILIM9)	16	R/W	7FF8h	25.3.17/ 546
E538	ADC High Limit Registers (ADC12_HILIM10)	16	R/W	7FF8h	25.3.17/ 546
E539	ADC High Limit Registers (ADC12_HILIM11)	16	R/W	7FF8h	25.3.17/ 546
E53A	ADC High Limit Registers (ADC12_HILIM12)	16	R/W	7FF8h	25.3.17/ 546
E53B	ADC High Limit Registers (ADC12_HILIM13)	16	R/W	7FF8h	25.3.17/ 546
E53C	ADC High Limit Registers (ADC12_HILIM14)	16	R/W	7FF8h	25.3.17/ 546
E53D	ADC High Limit Registers (ADC12_HILIM15)	16	R/W	7FF8h	25.3.17/ 546
E53E	ADC Offset Registers (ADC12_OFFST0)	16	R/W	0000h	25.3.18/ 546
E53F	ADC Offset Registers (ADC12_OFFST1)	16	R/W	0000h	25.3.18/ 546
E540	ADC Offset Registers (ADC12_OFFST2)	16	R/W	0000h	25.3.18/ 546
E541	ADC Offset Registers (ADC12_OFFST3)	16	R/W	0000h	25.3.18/ 546
E542	ADC Offset Registers (ADC12_OFFST4)	16	R/W	0000h	25.3.18/ 546
E543	ADC Offset Registers (ADC12_OFFST5)	16	R/W	0000h	25.3.18/ 546
E544	ADC Offset Registers (ADC12_OFFST6)	16	R/W	0000h	25.3.18/ 546
E545	ADC Offset Registers (ADC12_OFFST7)	16	R/W	0000h	25.3.18/ 546
E546	ADC Offset Registers (ADC12_OFFST8)	16	R/W	0000h	25.3.18/ 546
E547	ADC Offset Registers (ADC12_OFFST9)	16	R/W	0000h	25.3.18/ 546
E548	ADC Offset Registers (ADC12_OFFST10)	16	R/W	0000h	25.3.18/ 546
E549	ADC Offset Registers (ADC12_OFFST11)	16	R/W	0000h	25.3.18/ 546
E54A	ADC Offset Registers (ADC12_OFFST12)	16	R/W	0000h	25.3.18/ 546
E54B	ADC Offset Registers (ADC12_OFFST13)	16	R/W	0000h	25.3.18/ 546

Table continues on the next page...

ADC memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
E54C	ADC Offset Registers (ADC12_OFFST14)	16	R/W	0000h	25.3.18/ 546
E54D	ADC Offset Registers (ADC12_OFFST15)	16	R/W	0000h	25.3.18/ 546
E54E	ADC Power Control Register (ADC12_PWR)	16	R/W	1DA7h	25.3.19/ 547
E54F	ADC Calibration Register (ADC12_CAL)	16	R/W	0000h	25.3.20/ 550
E550	Gain Control 1 Register (ADC12_GC1)	16	R/W	0000h	25.3.21/ 551
E551	Gain Control 2 Register (ADC12_GC2)	16	R/W	0000h	25.3.22/ 552
E552	ADC Scan Control Register (ADC12_SCTRL)	16	R/W	0000h	25.3.23/ 554
E553	ADC Power Control Register (ADC12_PWR2)	16	R/W	0400h	25.3.24/ 555
E554	ADC Control Register 3 (ADC12_CTRL3)	16	R/W	0000h	25.3.25/ 556
E555	ADC Scan Halted Interrupt Enable Register (ADC12_SCHLTEN)	16	R/W	0000h	25.3.26/ 557

25.3.1 ADC Control Register 1 (ADCx_CTRL1)

Bits 14, 13, 12, and 11 in CTRL1 control all types of scans except parallel scans in the B converter when CTRL2[SIMULT]=0. Non-simultaneous parallel scan modes allow independent parallel scanning in the A and B converter. Bits 14, 13, 12, and 11 in CTRL2 are used to control B converter scans in non-simultaneous parallel scan modes.

Address: Base address + 0h offset

Bit	15	14	13	12	11	10	9	8
Read	DMAEN0	STOP0		SYNC0	EOSIE0	ZCIE	LLMTIE	HLMTIE
Write			START0					
Reset	0	1	0	1	0	0	0	0

Bit	7	6	5	4	3	2	1	0
Read	CHNCFG_L				0	SMODE		
Write								
Reset	0	0	0	0	0	1	0	1

ADCx_CTRL1 field descriptions

Field	Description
15 DMAEN0	DMA enable When this bit is asserted, the DMA source selected by CTRL3[DMASRC] causes the conversion results to be transferred by the DMA controller. 0 DMA is not enabled. 1 DMA is enabled.
14 STOP0	Stop When this bit is asserted, the current scan is stopped and no further scans can start. Any further SYNC0 input pulses (see CTRL1[SYNC0] bit) or writes to the CTRL1[START0] bit are ignored until this bit has been cleared. After the ADC is in stop mode, the results registers can be modified by the processor. Any changes to the result registers in stop mode are treated as if the analog core supplied the data. Therefore, limit checking, zero crossing, and associated interrupts can occur when authorized. **This is not the same as DSP STOP mode.** 0 Normal operation 1 Stop mode
13 START0	START0 Conversion A scan is started by writing 1 to this bit. This is a write only bit. Writing 1 to it again while the scan remains in process, is ignored. The ADC must be in a stable power configuration prior to writing the start bit. Refer to the functional description of power modes for further details. 0 No action 1 Start command is issued
12 SYNC0	SYNC0 Enable A conversion may be initiated by asserting a positive edge on the SYNC0 input. Any subsequent SYNC0 input pulses while the scan remains in process are ignored unless the scan is awaiting further SYNC inputs due to the SCTRL[SCn] bits. CTRL1[SYNC0] is cleared in ONCE mode, CTRL1[SMODE=000 or 001], when the first SYNC input is detected. This prevents unintentionally starting a new scan after the first scan has completed. The ADC must be in a stable power mode prior to SYNC0 input assertion. Refer to the functional description of power modes for further details. In "once" scan modes, only a first SYNC0 input pulse is honored. CTRL1[SYNC0] is cleared in this mode when the first SYNC input is detected. This prevents unintentionally starting a new scan after the first scan has completed. The CTRL1[SYNC0] bit can be set again at any time including while the scan remains in process 0 Scan is initiated by a write to CTRL1[START0] only 1 Use a SYNC0 input pulse or CTRL1[START0] to initiate a scan
11 EOSIE0	End Of Scan Interrupt Enable This bit enables an EOSI0 interrupt to be generated upon completion of the scan. For looping scan modes, the interrupt will trigger after the completion of each iteration of the loop. 0 **Interrupt disabled** 1 **Interrupt enabled**
10 ZCIE	Zero Crossing Interrupt Enable

Table continues on the next page...

ADCx_CTRL1 field descriptions (continued)

Field	Description
	This bit enables the zero crossing interrupt if the current result value has a sign change from the previous result as configured by the ZXCTRL1 and ZXCTRL2 registers. 0 **Interrupt disabled** 1 **Interrupt enabled**
9 LLMTIE	Low Limit Interrupt Enable This bit enables the Low Limit exceeded interrupt when the current result value is less than the low limit register value. The raw result value is compared to LOLIM[LLMT] before the offset register value is subtracted. 0 **Interrupt disabled** 1 **Interrupt enabled**
8 HLMTIE	High Limit Interrupt Enable This bit enables the High Limit exceeded interrupt if the current result value is greater than the high limit register value. The raw result value is compared to HILIM[HLMT] before the offset register value is subtracted. 0 **Interrupt disabled** 1 **Interrupt enabled**
7–4 CHNCFG_L	CHCNF (Channel Configure Low) bits The bits configure the analog inputs for either single ended or differential conversions. Differential measurements return the max value ((2**12)-1) when the + input is V_{REFH} and the - input is V_{REFLO}, return 0 when the + input is at V_{REFLO} and the - input is at V_{REFH}, and scale linearly between based on the voltage difference between the two signals. Single ended measurements return the max value when the input is at V_{REFH}, return 0 when the input is at V_{REFLO}, and scale linearly between based on the amount by which the input exceeds V_{REFLO}. xxx1 **Inputs = ANA0-ANA1** — Configured as differential pair (ANA0 is + and ANA1 is --) xxx0 **Inputs = ANA0-ANA1** — Both configured as single ended inputs xx1x **Inputs = ANA2-ANA3** — Configured as differential pair (ANA2 is + and ANA3 is --) xx0x **Inputs = ANA2-ANA3** — Both configured as single ended inputs x1xx **Inputs = ANB0-ANB1** — Configured as differential pair (ANB0 is + and ANB1 is --) x0xx **Inputs = ANB0-ANB1** — Both configured as single ended inputs 1xxx **Inputs = ANB2-ANB3** — Configured as differential pair (ANB2 is + and ANB3 is --) 0xxx **Inputs = ANB2-ANB3** — Both configured as single ended inputs
3 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
SMODE	ADC Scan Mode Control This field controls the ADC module's scan mode. All scan modes use 16 sample slots defined by the CLIST1-4 registers. A scan is the process of stepping through a subset of these sample slots, converting the input indicated by a slot, and storing the result. Unused slots may be disabled using the SDIS register. Input pairs ANA0-1, ANA2-3, ANA4-5, ANA6-7, ANB0-1, ANB2-3, ANB4-5, and ANB6-7 may be configured as differential pairs using the CHNCFG fields. When a slot refers to either member of a differential pair, a differential measurement on that pair is made; otherwise, a single ended measurement is taken on that input. The CTRL*[CHNCFG] fields' descriptions detail differential and single ended measurement. The SMODE field determines whether the slots are used to perform one long sequential scan or two shorter parallel scans, each performed by one of the two converters. SMODE controls how

Table continues on the next page...

ADCx_CTRL1 field descriptions (continued)

Field	Description
	these scans are initiated and terminated. It also controls whether the scans are performed once or repetitively. For details, refer to Sequential Versus Parallel Sampling and Scan Sequencing . Parallel scans may be simultaneous (CTRL2[SIMULT] is 1) or non-simultaneous. Simultaneous parallel scans perform the A and B converter scan in lock step using one set of shared controls. Non-simultaneous parallel scans operate the A and B converters independently, with each converter using its own set of controls. Refer to the CTRL2[SIMULT] bit's description for details. Setting any sequential mode overrides the setting of CTRL2[SIMULT]. 000 **Once (single) sequential** — Upon start or an enabled sync signal, samples are taken one at a time starting with CLIST1[SAMPLE0], until the first disabled sample is encountered. If no disabled sample is encountered, conversion concludes after CLIST4[SAMPLE15]. If the scan is initiated by a SYNC signal, only one scan is completed because the CTRL*[SYNC*] bit is cleared automatically by the initial SYNC detection. CTRL*[SYNC*] can be set again at any time during the scan. 001 **Once parallel** — Upon start or an armed and enabled sync signal: In parallel, converter A converts SAMPLEs 0-7, and converter B converts SAMPLEs 8-15. When CTRL2[SIMULT] is 1 (default), scanning stops when either converter encounters a disabled sample or both converters complete all 8 samples. When CTRL2[SIMULT] is 0, a converter stops scanning when it encounters a disabled sample or completes all 8 samples. If the scan is initiated by a SYNC signal, only one scan is completed because the CTRL*[SYNC*] bit is cleared automatically by the initial SYNC detection. CTRL*[SYNC*] can be set again at any time during the scan. If CTRL2[SIMULT] is 0, the B converter must be rearmed by writing the CTRL2[SYNC1] bit. 010 **Loop sequential** — Upon an initial start or enabled sync pulse, up to 16 samples in the order SAMPLEs 0-15 are taken one at a time until a disabled sample is encountered. The process repeats perpetually until the CTRL1[STOP0] bit is set. While a loop mode is running, any additional start commands or sync pulses are ignored unless the scan is paused using the SCTRL[SC*] bits. If PWR[ASB] or PWR[APD] is the selected power mode control, PWR[PUDELAY] is applied only on the first conversion. 011 **Loop parallel** — Upon an initial start or enabled sync pulse, converter A converts SAMPLEs 0-7, and converter B converts SAMPLEs 8-15. Each time a converter completes its current scan, it immediately restarts its scan sequence. This process continues until the CTRL*[STOP*] bit is asserted. While a loop is running, any additional start commands or sync pulses are ignored unless the scan is paused using the SCTRL[SC*] bits. When CTRL2[SIMULT] is 1 (default), scanning restarts when either converter encounters a disabled sample. When CTRL2[SIMULT] is 0, a converter restarts scanning when it encounters a disabled sample. If PWR[ASB] or PWR[APD] is the selected power mode control, PWR[PUDELAY] is applied only on the first conversion. 100 **Triggered sequential** — Upon start or an enabled sync signal, samples are taken one at a time starting with CLIST1[SAMPLE0], until the first disabled sample is encountered. If no disabled sample is encountered, conversion concludes after CLIST4[SAMPLE15]. If external sync is enabled, new scans start for each SYNC pulse that does not overlap with a current scan in progress. 101 **Triggered parallel (default)** — Upon start or an enabled sync signal: In parallel, converter A converts SAMPLEs 0-7, and converter B converts SAMPLEs 8-15. When CTRL2[SIMULT] is 1 (default), scanning stops when either converter encounters a disabled sample. When CTRL2[SIMULT] is 0, a converter stops scanning when it encounters a disabled sample. If external sync is enabled, new scans start for each SYNC pulse that does not overlap with a current scan in progress. 11x **Reserved**

25.3.2 ADC Control Register 2 (ADC*x*_CTRL2)

Address: Base address + 1h offset

Bit	15	14	13	12	11	10	9	8
Read	DMAEN1	STOP1	0	SYNC1	EOSIE1	CHNCFG_H		
Write			START1					
Reset	0	1	0	1	0	0	0	0

Bit	7	6	5	4	3	2	1	0
Read	CHNCFG_H	SIMULT	DIV0					
Write								
Reset	0	1	0	0	0	1	0	0

ADCx_CTRL2 field descriptions

Field	Description
15 DMAEN1	DMA enable During parallel scan modes when SIMULT=0, this bit enables DMA for converter B. When this bit is asserted, the DMA source selected by CTRL3[DMASRC] causes the conversion results to be transferred by the DMA controller. 0 DMA is not enabled. 1 DMA is enabled.
14 STOP1	Stop During parallel scan modes when SIMULT = 0, this bit enables stop control of a B converter parallel scan. When this bit is asserted, the current scan is stopped and no further scans can start. Any further SYNC1 input pulses (see CTRL2[SYNC1] bit) or writes to the CTRL2[START1] bit are ignored until this bit has been cleared. After the ADC is in stop mode, the results registers can be modified by the processor. Any changes to the result registers in stop mode are treated as if the analog core supplied the data. Therefore, limit checking, zero crossing, and associated interrupts can occur when authorized. **This is not the same as DSP STOP mode.** 0 Normal operation 1 Stop mode
13 START1	START1 Conversion During parallel scan modes when SIMULT = 0, this bit enables start control of a B converter parallel scan. A scan is started by writing 1 to this bit. This is a write only bit. Writing 1 to it again while the scan remains in process, is ignored. The ADC must be in a stable power configuration prior to writing the start bit. Refer to the functional description of power modes for further details. 0 No action 1 Start command is issued
12 SYNC1	SYNC1 Enable

Table continues on the next page...

ADCx_CTRL2 field descriptions (continued)

Field	Description
	During parallel scan modes when CTRL2[SIMULT]=0, setting this bit to 1 permits a B converter parallel scan to be initiated by asserting the SYNC1 input for at least one ADC clock cycle. CTRL2[SYNC1] is cleared in ONCE mode, CTRL1[SMODE=000 or 001], when the first SYNC input is detected. This prevents unintentionally starting a new scan after the first scan has completed. The ADC must be in a stable power mode prior to SYNC1 input assertion. Refer to the functional description of power modes for further details. In "once" scan modes, only a first SYNC1 input pulse is honored. CTRL2[SYNC1] is cleared in this mode when the first SYNC1 input is detected. This prevents unintentionally starting a new scan after the first scan has completed. The CTRL2[SYNC1] bit can be set again at any time including while the scan remains in process. 0 B converter parallel scan is initiated by a write to CTRL2[START1] bit only 1 Use a SYNC1 input pulse or CTRL2[START1] bit to initiate a B converter parallel scan
11 EOSIE1	End Of Scan Interrupt Enable During parallel scan modes when SIMULT = 0, this bit enables interrupt control for a B converter parallel scan. This bit enables an EOSI1 interrupt to be generated upon completion of the scan. For looping scan modes, the interrupt will trigger after the completion of each iteration of the loop. 0 **Interrupt disabled** 1 **Interrupt enabled**
10–7 CHNCFG_H	CHCNF (Channel Configure High) bits The bits configure the analog inputs for either single ended or differential conversions. Differential measurements return the max value ((2**12)-1) when the + input is V_{REFH} and the - input is V_{REFLO}, return 0 when the + input is at V_{REFLO} and the - input is at V_{REFH}, and scale linearly between based on the voltage difference between the two signals. Single ended measurements return the max value when the input is at V_{REFH}, return 0 when the input is at V_{REFLO}, and scale linearly between based on the amount by which the input exceeds V_{REFLO}. xxx1 **Inputs = ANA4-ANA5** — Configured as differential pair (ANA4 is + and ANA5 is --) xxx0 **Inputs = ANA4-ANA5** — Both configured as single ended inputs xx1x **Inputs = ANA6-ANA7** — Configured as differential pair (ANA6 is + and ANA7 is --) xx0x **Inputs = ANA6-ANA7** — Both configured as single ended inputs x1xx **Inputs = ANB4-ANB5** — Configured as differential pair (ANB4 is + and ANB5 is --) x0xx **Inputs = ANB4-ANB5** — Both configured as single ended inputs 1xxx **Inputs = ANB6-ANB7** — Configured as differential pair (ANB6 is + and ANB7 is --) 0xxx **Inputs = ANB6-ANB7** — Both configured as single ended inputs
6 SIMULT	Simultaneous mode This bit only affects parallel scan modes. By default (CTRL2[SIMULT]=1) parallel scans operate in simultaneous mode. The scans in the A and B converter operate simultaneously and always result in pairs of simultaneous conversions in the A and B converter. CTRL1[STOP0, SYNC0, and START0] control bits and the SYNC0 input are used to start and stop scans in both converters simultaneously. A scan ends in both converters when either converter encounters a disabled sample slot. When the parallel scan completes, the STAT[EOSI0] triggers if CTRL1[EOSIE0] is set. The STAT[CIP0] status bit indicates that a parallel scan is in process. When CTRL2[SIMULT]=0, parallel scans in the A and B converters operate independently. The B converter has its own independent set of the above controls (with a 1 suffix) which control its operation and report its status. Each converter's scan continues until its sample list is exhausted (8 samples) or a

Table continues on the next page...

ADCx_CTRL2 field descriptions (continued)

Field	Description
	disabled sample IN ITS LIST is encountered. For looping parallel scan mode, each converter starts its next iteration as soon as the previous iteration in that converter is complete and continues until the CTRL*[STOP*] bit for that converter is asserted. 0 Parallel scans done independently 1 Parallel scans done simultaneously (default)
DIV0	Clock Divisor Select The divider circuit generates the ADC clock by dividing the system clock: • When DIV0 is 0, the divisor is 2. • For all other DIV0 values, the divisor is 1 more than the decimal value of DIV0: (DIV0)Â +Â 1d. A DIV0 value must be chosen so the ADC clock does not exceed the maximum frequency. This clock is used by ADCA during all scans and is used by ADCB during sequential scan modes and during parallel simultaneous scan modes. The following table shows the ADC clock frequency based on the value of DIV0 for various OCCS clock source configurations.

DIV0	Divisor	ROSC Standby 400kHz	ROSC Normal 8MHz	PLL 100 MHz	External CLK
00_0000	2	100K	2.00M	50.0M	CLK/4
00_0001	2	100K	2.00M	50.0M	CLK/4
00_0010	3	100K	1.33M	33.33M	CLK/6
00_0011	4	100K	1.00M	25.0M	CLK/8
00_0100	5	100K	800K	20.00M	CLK/10
00_0101	6	100K	667K	16.67M	CLK/12
-	-		-	-	
-	-		-	-	
11_1111	64	100K	62.5K	1.56M	CLK/128

25.3.3 ADC Zero Crossing Control 1 Register (ADCx_ZXCTRL1)

Address: Base address + 2h offset

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read Write	ZCE7		ZCE6		ZCE5		ZCE4		ZCE3		ZCE2		ZCE1		ZCE0	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ADCx_ZXCTRL1 field descriptions

Field	Description
15–14 ZCE7	Zero crossing enable 7 00 **Zero Crossing disabled**

Table continues on the next page...

ADCx_ZXCTRL1 field descriptions (continued)

Field	Description
	01 **Zero Crossing enabled for positive to negative sign change** 10 **Zero Crossing enabled for negative to positive sign change** 11 **Zero Crossing enabled for any sign change**
13–12 ZCE6	Zero crossing enable 6 00 **Zero Crossing disabled** 01 **Zero Crossing enabled for positive to negative sign change** 10 **Zero Crossing enabled for negative to positive sign change** 11 **Zero Crossing enabled for any sign change**
11–10 ZCE5	Zero crossing enable 5 00 **Zero Crossing disabled** 01 **Zero Crossing enabled for positive to negative sign change** 10 **Zero Crossing enabled for negative to positive sign change** 11 **Zero Crossing enabled for any sign change**
9–8 ZCE4	Zero crossing enable 4 00 **Zero Crossing disabled** 01 **Zero Crossing enabled for positive to negative sign change** 10 **Zero Crossing enabled for negative to positive sign change** 11 **Zero Crossing enabled for any sign change**
7–6 ZCE3	Zero crossing enable 3 00 **Zero Crossing disabled** 01 **Zero Crossing enabled for positive to negative sign change** 10 **Zero Crossing enabled for negative to positive sign change** 11 **Zero Crossing enabled for any sign change**
5–4 ZCE2	Zero crossing enable 2 00 **Zero Crossing disabled** 01 **Zero Crossing enabled for positive to negative sign change** 10 **Zero Crossing enabled for negative to positive sign change** 11 **Zero Crossing enabled for any sign change**
3–2 ZCE1	Zero crossing enable 1 00 **Zero Crossing disabled** 01 **Zero Crossing enabled for positive to negative sign change** 10 **Zero Crossing enabled for negative to positive sign change** 11 **Zero Crossing enabled for any sign change**
ZCE0	Zero crossing enable 0 00 **Zero Crossing disabled** 01 **Zero Crossing enabled for positive to negative sign change** 10 **Zero Crossing enabled for negative to positive sign change** 11 **Zero Crossing enabled for any sign change**

25.3.4 ADC Zero Crossing Control 2 Register (ADCx_ZXCTRL2)

Address: Base address + 3h offset

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read Write	ZCE15		ZCE14		ZCE13		ZCE12		ZCE11		ZCE10		ZCE9		ZCE8	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ADCx_ZXCTRL2 field descriptions

Field	Description
15–14 ZCE15	Zero crossing enable 15 00 **Zero Crossing disabled** 01 **Zero Crossing enabled for positive to negative sign change** 10 **Zero Crossing enabled for negative to positive sign change** 11 **Zero Crossing enabled for any sign change**
13–12 ZCE14	Zero crossing enable 14 00 **Zero Crossing disabled** 01 **Zero Crossing enabled for positive to negative sign change** 10 **Zero Crossing enabled for negative to positive sign change** 11 **Zero Crossing enabled for any sign change**
11–10 ZCE13	Zero crossing enable 13 00 **Zero Crossing disabled** 01 **Zero Crossing enabled for positive to negative sign change** 10 **Zero Crossing enabled for negative to positive sign change** 11 **Zero Crossing enabled for any sign change**
9–8 ZCE12	Zero crossing enable 12 00 **Zero Crossing disabled** 01 **Zero Crossing enabled for positive to negative sign change** 10 **Zero Crossing enabled for negative to positive sign change** 11 **Zero Crossing enabled for any sign change**
7–6 ZCE11	Zero crossing enable 11 00 **Zero Crossing disabled** 01 **Zero Crossing enabled for positive to negative sign change** 10 **Zero Crossing enabled for negative to positive sign change** 11 **Zero Crossing enabled for any sign change**
5–4 ZCE10	Zero crossing enable 10 00 **Zero Crossing disabled** 01 **Zero Crossing enabled for positive to negative sign change** 10 **Zero Crossing enabled for negative to positive sign change** 11 **Zero Crossing enabled for any sign change**
3–2 ZCE9	Zero crossing enable 9

Table continues on the next page...

ADCx_ZXCTRL2 field descriptions (continued)

Field	Description
	00 **Zero Crossing disabled** 01 **Zero Crossing enabled for positive to negative sign change** 10 **Zero Crossing enabled for negative to positive sign change** 11 **Zero Crossing enabled for any sign change**
ZCE8	Zero crossing enable 8 00 **Zero Crossing disabled** 01 **Zero Crossing enabled for positive to negative sign change** 10 **Zero Crossing enabled for negative to positive sign change** 11 **Zero Crossing enabled for any sign change**

25.3.5 ADC Channel List Register 1 (ADCx_CLIST1)

Address: Base address + 4h offset

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read Write	SAMPLE3				SAMPLE2				SAMPLE1				SAMPLE0			
Reset	0	0	1	1	0	0	1	0	0	0	0	1	0	0	0	0

ADCx_CLIST1 field descriptions

Field	Description
15–12 SAMPLE3	Sample Field 3 0000 **Single Ended: ANA0, Differential: ANA0+, ANA1-** 0001 **Single Ended: ANA1, Differential: ANA0+, ANA1-** 0010 **Single Ended: ANA2, Differential: ANA2+, ANA3-** 0011 **Single Ended: ANA3, Differential: ANA2+, ANA3-** 0100 **Single Ended: ANA4, Differential: ANA4+, ANA5-** 0101 **Single Ended: ANA5, Differential: ANA4+, ANA5-** 0110 **Single Ended: ANA6, Differential: ANA6+, ANA7-** 0111 **Single Ended: ANA7, Differential: ANA6+, ANA7-** 1000 **Single Ended: ANB0, Differential: ANB0+, ANB1-** 1001 **Single Ended: ANB1, Differential: ANB0+, ANB1-** 1010 **Single Ended: ANB2, Differential: ANB2+, ANB3-** 1011 **Single Ended: ANB3, Differential: ANB2+, ANB3-** 1100 **Single Ended: ANB4, Differential: ANB4+, ANB5-** 1101 **Single Ended: ANB5, Differential: ANB4+, ANB5-** 1110 **Single Ended: ANB6, Differential: ANB6+, ANB7-** 1111 **Single Ended: ANB7, Differential: ANB6+, ANB7-**
11–8 SAMPLE2	Sample Field 2 0000 **Single Ended: ANA0, Differential: ANA0+, ANA1-** 0001 **Single Ended: ANA1, Differential: ANA0+, ANA1-** 0010 **Single Ended: ANA2, Differential: ANA2+, ANA3-**

Table continues on the next page...

ADCx_CLIST1 field descriptions (continued)

Field	Description
	0011 **Single Ended: ANA3, Differential: ANA2+, ANA3-** 0100 **Single Ended: ANA4, Differential: ANA4+, ANA5-** 0101 **Single Ended: ANA5, Differential: ANA4+, ANA5-** 0110 **Single Ended: ANA6, Differential: ANA6+, ANA7-** 0111 **Single Ended: ANA7, Differential: ANA6+, ANA7-** 1000 **Single Ended: ANB0, Differential: ANB0+, ANB1-** 1001 **Single Ended: ANB1, Differential: ANB0+, ANB1-** 1010 **Single Ended: ANB2, Differential: ANB2+, ANB3-** 1011 **Single Ended: ANB3, Differential: ANB2+, ANB3-** 1100 **Single Ended: ANB4, Differential: ANB4+, ANB5-** 1101 **Single Ended: ANB5, Differential: ANB4+, ANB5-** 1110 **Single Ended: ANB6, Differential: ANB6+, ANB7-** 1111 **Single Ended: ANB7, Differential: ANB6+, ANB7-**
7–4 SAMPLE1	Sample Field 1 0000 **Single Ended: ANA0, Differential: ANA0+, ANA1-** 0001 **Single Ended: ANA1, Differential: ANA0+, ANA1-** 0010 **Single Ended: ANA2, Differential: ANA2+, ANA3-** 0011 **Single Ended: ANA3, Differential: ANA2+, ANA3-** 0100 **Single Ended: ANA4, Differential: ANA4+, ANA5-** 0101 **Single Ended: ANA5, Differential: ANA4+, ANA5-** 0110 **Single Ended: ANA6, Differential: ANA6+, ANA7-** 0111 **Single Ended: ANA7, Differential: ANA6+, ANA7-** 1000 **Single Ended: ANB0, Differential: ANB0+, ANB1-** 1001 **Single Ended: ANB1, Differential: ANB0+, ANB1-** 1010 **Single Ended: ANB2, Differential: ANB2+, ANB3-** 1011 **Single Ended: ANB3, Differential: ANB2+, ANB3-** 1100 **Single Ended: ANB4, Differential: ANB4+, ANB5-** 1101 **Single Ended: ANB5, Differential: ANB4+, ANB5-** 1110 **Single Ended: ANB6, Differential: ANB6+, ANB7-** 1111 **Single Ended: ANB7, Differential: ANB6+, ANB7-**
SAMPLE0	Sample Field 0 0000 **Single Ended: ANA0, Differential: ANA0+, ANA1-** 0001 **Single Ended: ANA1, Differential: ANA0+, ANA1-** 0010 **Single Ended: ANA2, Differential: ANA2+, ANA3-** 0011 **Single Ended: ANA3, Differential: ANA2+, ANA3-** 0100 **Single Ended: ANA4, Differential: ANA4+, ANA5-** 0101 **Single Ended: ANA5, Differential: ANA4+, ANA5-** 0110 **Single Ended: ANA6, Differential: ANA6+, ANA7-** 0111 **Single Ended: ANA7, Differential: ANA6+, ANA7-** 1000 **Single Ended: ANB0, Differential: ANB0+, ANB1-** 1001 **Single Ended: ANB1, Differential: ANB0+, ANB1-** 1010 **Single Ended: ANB2, Differential: ANB2+, ANB3-** 1011 **Single Ended: ANB3, Differential: ANB2+, ANB3-** 1100 **Single Ended: ANB4, Differential: ANB4+, ANB5-** 1101 **Single Ended: ANB5, Differential: ANB4+, ANB5-**

Table continues on the next page...

ADCx_CLIST1 field descriptions (continued)

Field	Description
	1110 **Single Ended: ANB6, Differential: ANB6+, ANB7-** 1111 **Single Ended: ANB7, Differential: ANB6+, ANB7-**

25.3.6 ADC Channel List Register 2 (ADCx_CLIST2)

Address: Base address + 5h offset

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read Write	SAMPLE7				SAMPLE6				SAMPLE5				SAMPLE4			
Reset	0	1	1	1	0	1	1	0	0	1	0	1	0	1	0	0

ADCx_CLIST2 field descriptions

Field	Description
15–12 SAMPLE7	Sample Field 7 0000 **Single Ended: ANA0, Differential: ANA0+, ANA1-** 0001 **Single Ended: ANA1, Differential: ANA0+, ANA1-** 0010 **Single Ended: ANA2, Differential: ANA2+, ANA3-** 0011 **Single Ended: ANA3, Differential: ANA2+, ANA3-** 0100 **Single Ended: ANA4, Differential: ANA4+, ANA5-** 0101 **Single Ended: ANA5, Differential: ANA4+, ANA5-** 0110 **Single Ended: ANA6, Differential: ANA6+, ANA7-** 0111 **Single Ended: ANA7, Differential: ANA6+, ANA7-** 1000 **Single Ended: ANB0, Differential: ANB0+, ANB1-** 1001 **Single Ended: ANB1, Differential: ANB0+, ANB1-** 1010 **Single Ended: ANB2, Differential: ANB2+, ANB3-** 1011 **Single Ended: ANB3, Differential: ANB2+, ANB3-** 1100 **Single Ended: ANB4, Differential: ANB4+, ANB5-** 1101 **Single Ended: ANB5, Differential: ANB4+, ANB5-** 1110 **Single Ended: ANB6, Differential: ANB6+, ANB7-** 1111 **Single Ended: ANB7, Differential: ANB6+, ANB7-**
11–8 SAMPLE6	Sample Field 6 0000 **Single Ended: ANA0, Differential: ANA0+, ANA1-** 0001 **Single Ended: ANA1, Differential: ANA0+, ANA1-** 0010 **Single Ended: ANA2, Differential: ANA2+, ANA3-** 0011 **Single Ended: ANA3, Differential: ANA2+, ANA3-** 0100 **Single Ended: ANA4, Differential: ANA4+, ANA5-** 0101 **Single Ended: ANA5, Differential: ANA4+, ANA5-** 0110 **Single Ended: ANA6, Differential: ANA6+, ANA7-** 0111 **Single Ended: ANA7, Differential: ANA6+, ANA7-** 1000 **Single Ended: ANB0, Differential: ANB0+, ANB1-** 1001 **Single Ended: ANB1, Differential: ANB0+, ANB1-** 1010 **Single Ended: ANB2, Differential: ANB2+, ANB3-**

Table continues on the next page...

ADCx_CLIST2 field descriptions (continued)

Field	Description
	1011 **Single Ended: ANB3, Differential: ANB2+, ANB3-** 1100 **Single Ended: ANB4, Differential: ANB4+, ANB5-** 1101 **Single Ended: ANB5, Differential: ANB4+, ANB5-** 1110 **Single Ended: ANB6, Differential: ANB6+, ANB7-** 1111 **Single Ended: ANB7, Differential: ANB6+, ANB7-**
7–4 SAMPLE5	Sample Field 5 0000 **Single Ended: ANA0, Differential: ANA0+, ANA1-** 0001 **Single Ended: ANA1, Differential: ANA0+, ANA1-** 0010 **Single Ended: ANA2, Differential: ANA2+, ANA3-** 0011 **Single Ended: ANA3, Differential: ANA2+, ANA3-** 0100 **Single Ended: ANA4, Differential: ANA4+, ANA5-** 0101 **Single Ended: ANA5, Differential: ANA4+, ANA5-** 0110 **Single Ended: ANA6, Differential: ANA6+, ANA7-** 0111 **Single Ended: ANA7, Differential: ANA6+, ANA7-** 1000 **Single Ended: ANB0, Differential: ANB0+, ANB1-** 1001 **Single Ended: ANB1, Differential: ANB0+, ANB1-** 1010 **Single Ended: ANB2, Differential: ANB2+, ANB3-** 1011 **Single Ended: ANB3, Differential: ANB2+, ANB3-** 1100 **Single Ended: ANB4, Differential: ANB4+, ANB5-** 1101 **Single Ended: ANB5, Differential: ANB4+, ANB5-** 1110 **Single Ended: ANB6, Differential: ANB6+, ANB7-** 1111 **Single Ended: ANB7, Differential: ANB6+, ANB7-**
SAMPLE4	Sample Field 4 0000 **Single Ended: ANA0, Differential: ANA0+, ANA1-** 0001 **Single Ended: ANA1, Differential: ANA0+, ANA1-** 0010 **Single Ended: ANA2, Differential: ANA2+, ANA3-** 0011 **Single Ended: ANA3, Differential: ANA2+, ANA3-** 0100 **Single Ended: ANA4, Differential: ANA4+, ANA5-** 0101 **Single Ended: ANA5, Differential: ANA4+, ANA5-** 0110 **Single Ended: ANA6, Differential: ANA6+, ANA7-** 0111 **Single Ended: ANA7, Differential: ANA6+, ANA7-** 1000 **Single Ended: ANB0, Differential: ANB0+, ANB1-** 1001 **Single Ended: ANB1, Differential: ANB0+, ANB1-** 1010 **Single Ended: ANB2, Differential: ANB2+, ANB3-** 1011 **Single Ended: ANB3, Differential: ANB2+, ANB3-** 1100 **Single Ended: ANB4, Differential: ANB4+, ANB5-** 1101 **Single Ended: ANB5, Differential: ANB4+, ANB5-** 1110 **Single Ended: ANB6, Differential: ANB6+, ANB7-** 1111 **Single Ended: ANB7, Differential: ANB6+, ANB7-**

25.3.7 ADC Channel List Register 3 (ADC*x*_CLIST3)

Address: Base address + 6h offset

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read Write	SAMPLE11				SAMPLE10				SAMPLE9				SAMPLE8			
Reset	1	0	1	1	1	0	1	0	1	0	0	1	1	0	0	0

ADC*x*_CLIST3 field descriptions

Field	Description
15–12 SAMPLE11	Sample Field 11 0000 **Single Ended: ANA0, Differential: ANA0+, ANA1-** 0001 **Single Ended: ANA1, Differential: ANA0+, ANA1-** 0010 **Single Ended: ANA2, Differential: ANA2+, ANA3-** 0011 **Single Ended: ANA3, Differential: ANA2+, ANA3-** 0100 **Single Ended: ANA4, Differential: ANA4+, ANA5-** 0101 **Single Ended: ANA5, Differential: ANA4+, ANA5-** 0110 **Single Ended: ANA6, Differential: ANA6+, ANA7-** 0111 **Single Ended: ANA7, Differential: ANA6+, ANA7-** 1000 **Single Ended: ANB0, Differential: ANB0+, ANB1-** 1001 **Single Ended: ANB1, Differential: ANB0+, ANB1-** 1010 **Single Ended: ANB2, Differential: ANB2+, ANB3-** 1011 **Single Ended: ANB3, Differential: ANB2+, ANB3-** 1100 **Single Ended: ANB4, Differential: ANB4+, ANB5-** 1101 **Single Ended: ANB5, Differential: ANB4+, ANB5-** 1110 **Single Ended: ANB6, Differential: ANB6+, ANB7-** 1111 **Single Ended: ANB7, Differential: ANB6+, ANB7-**
11–8 SAMPLE10	Sample Field 10 0000 **Single Ended: ANA0, Differential: ANA0+, ANA1-** 0001 **Single Ended: ANA1, Differential: ANA0+, ANA1-** 0010 **Single Ended: ANA2, Differential: ANA2+, ANA3-** 0011 **Single Ended: ANA3, Differential: ANA2+, ANA3-** 0100 **Single Ended: ANA4, Differential: ANA4+, ANA5-** 0101 **Single Ended: ANA5, Differential: ANA4+, ANA5-** 0110 **Single Ended: ANA6, Differential: ANA6+, ANA7-** 0111 **Single Ended: ANA7, Differential: ANA6+, ANA7-** 1000 **Single Ended: ANB0, Differential: ANB0+, ANB1-** 1001 **Single Ended: ANB1, Differential: ANB0+, ANB1-** 1010 **Single Ended: ANB2, Differential: ANB2+, ANB3-** 1011 **Single Ended: ANB3, Differential: ANB2+, ANB3-** 1100 **Single Ended: ANB4, Differential: ANB4+, ANB5-** 1101 **Single Ended: ANB5, Differential: ANB4+, ANB5-** 1110 **Single Ended: ANB6, Differential: ANB6+, ANB7-** 1111 **Single Ended: ANB7, Differential: ANB6+, ANB7-**

Table continues on the next page...

ADCx_CLIST3 field descriptions (continued)

Field	Description
7–4 SAMPLE9	Sample Field 9 0000 **Single Ended: ANA0, Differential: ANA0+, ANA1-** 0001 **Single Ended: ANA1, Differential: ANA0+, ANA1-** 0010 **Single Ended: ANA2, Differential: ANA2+, ANA3-** 0011 **Single Ended: ANA3, Differential: ANA2+, ANA3-** 0100 **Single Ended: ANA4, Differential: ANA4+, ANA5-** 0101 **Single Ended: ANA5, Differential: ANA4+, ANA5-** 0110 **Single Ended: ANA6, Differential: ANA6+, ANA7-** 0111 **Single Ended: ANA7, Differential: ANA6+, ANA7-** 1000 **Single Ended: ANB0, Differential: ANB0+, ANB1-** 1001 **Single Ended: ANB1, Differential: ANB0+, ANB1-** 1010 **Single Ended: ANB2, Differential: ANB2+, ANB3-** 1011 **Single Ended: ANB3, Differential: ANB2+, ANB3-** 1100 **Single Ended: ANB4, Differential: ANB4+, ANB5-** 1101 **Single Ended: ANB5, Differential: ANB4+, ANB5-** 1110 **Single Ended: ANB6, Differential: ANB6+, ANB7-** 1111 **Single Ended: ANB7, Differential: ANB6+, ANB7-**
SAMPLE8	Sample Field 8 0000 **Single Ended: ANA0, Differential: ANA0+, ANA1-** 0001 **Single Ended: ANA1, Differential: ANA0+, ANA1-** 0010 **Single Ended: ANA2, Differential: ANA2+, ANA3-** 0011 **Single Ended: ANA3, Differential: ANA2+, ANA3-** 0100 **Single Ended: ANA4, Differential: ANA4+, ANA5-** 0101 **Single Ended: ANA5, Differential: ANA4+, ANA5-** 0110 **Single Ended: ANA6, Differential: ANA6+, ANA7-** 0111 **Single Ended: ANA7, Differential: ANA6+, ANA7-** 1000 **Single Ended: ANB0, Differential: ANB0+, ANB1-** 1001 **Single Ended: ANB1, Differential: ANB0+, ANB1-** 1010 **Single Ended: ANB2, Differential: ANB2+, ANB3-** 1011 **Single Ended: ANB3, Differential: ANB2+, ANB3-** 1100 **Single Ended: ANB4, Differential: ANB4+, ANB5-** 1101 **Single Ended: ANB5, Differential: ANB4+, ANB5-** 1110 **Single Ended: ANB6, Differential: ANB6+, ANB7-** 1111 **Single Ended: ANB7, Differential: ANB6+, ANB7-**

25.3.8 ADC Channel List Register 4 (ADCx_CLIST4)

Address: Base address + 7h offset

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read Write	SAMPLE15				SAMPLE14				SAMPLE13				SAMPLE12			
Reset	1	1	1	1	1	1	1	0	1	1	0	1	1	1	0	0

ADCx_CLIST4 field descriptions

Field	Description
15–12 SAMPLE15	Sample Field 15 0000 **Single Ended: ANA0, Differential: ANA0+, ANA1-** 0001 **Single Ended: ANA1, Differential: ANA0+, ANA1-** 0010 **Single Ended: ANA2, Differential: ANA2+, ANA3-** 0011 **Single Ended: ANA3, Differential: ANA2+, ANA3-** 0100 **Single Ended: ANA4, Differential: ANA4+, ANA5-** 0101 **Single Ended: ANA5, Differential: ANA4+, ANA5-** 0110 **Single Ended: ANA6, Differential: ANA6+, ANA7-** 0111 **Single Ended: ANA7, Differential: ANA6+, ANA7-** 1000 **Single Ended: ANB0, Differential: ANB0+, ANB1-** 1001 **Single Ended: ANB1, Differential: ANB0+, ANB1-** 1010 **Single Ended: ANB2, Differential: ANB2+, ANB3-** 1011 **Single Ended: ANB3, Differential: ANB2+, ANB3-** 1100 **Single Ended: ANB4, Differential: ANB4+, ANB5-** 1101 **Single Ended: ANB5, Differential: ANB4+, ANB5-** 1110 **Single Ended: ANB6, Differential: ANB6+, ANB7-** 1111 **Single Ended: ANB7, Differential: ANB6+, ANB7-**
11–8 SAMPLE14	Sample Field 14 0000 **Single Ended: ANA0, Differential: ANA0+, ANA1-** 0001 **Single Ended: ANA1, Differential: ANA0+, ANA1-** 0010 **Single Ended: ANA2, Differential: ANA2+, ANA3-** 0011 **Single Ended: ANA3, Differential: ANA2+, ANA3-** 0100 **Single Ended: ANA4, Differential: ANA4+, ANA5-** 0101 **Single Ended: ANA5, Differential: ANA4+, ANA5-** 0110 **Single Ended: ANA6, Differential: ANA6+, ANA7-** 0111 **Single Ended: ANA7, Differential: ANA6+, ANA7-** 1000 **Single Ended: ANB0, Differential: ANB0+, ANB1-** 1001 **Single Ended: ANB1, Differential: ANB0+, ANB1-** 1010 **Single Ended: ANB2, Differential: ANB2+, ANB3-** 1011 **Single Ended: ANB3, Differential: ANB2+, ANB3-** 1100 **Single Ended: ANB4, Differential: ANB4+, ANB5-** 1101 **Single Ended: ANB5, Differential: ANB4+, ANB5-** 1110 **Single Ended: ANB6, Differential: ANB6+, ANB7-** 1111 **Single Ended: ANB7, Differential: ANB6+, ANB7-**
7–4 SAMPLE13	Sample Field 13 0000 **Single Ended: ANA0, Differential: ANA0+, ANA1-** 0001 **Single Ended: ANA1, Differential: ANA0+, ANA1-** 0010 **Single Ended: ANA2, Differential: ANA2+, ANA3-** 0011 **Single Ended: ANA3, Differential: ANA2+, ANA3-** 0100 **Single Ended: ANA4, Differential: ANA4+, ANA5-** 0101 **Single Ended: ANA5, Differential: ANA4+, ANA5-** 0110 **Single Ended: ANA6, Differential: ANA6+, ANA7-** 0111 **Single Ended: ANA7, Differential: ANA6+, ANA7-** 1000 **Single Ended: ANB0, Differential: ANB0+, ANB1-** 1001 **Single Ended: ANB1, Differential: ANB0+, ANB1-**

Table continues on the next page...

ADCx_CLIST4 field descriptions (continued)

Field	Description
	1010 **Single Ended: ANB2, Differential: ANB2+, ANB3-** 1011 **Single Ended: ANB3, Differential: ANB2+, ANB3-** 1100 **Single Ended: ANB4, Differential: ANB4+, ANB5-** 1101 **Single Ended: ANB5, Differential: ANB4+, ANB5-** 1110 **Single Ended: ANB6, Differential: ANB6+, ANB7-** 1111 **Single Ended: ANB7, Differential: ANB6+, ANB7-**
SAMPLE12	Sample Field 12 0000 **Single Ended: ANA0, Differential: ANA0+, ANA1-** 0001 **Single Ended: ANA1, Differential: ANA0+, ANA1-** 0010 **Single Ended: ANA2, Differential: ANA2+, ANA3-** 0011 **Single Ended: ANA3, Differential: ANA2+, ANA3-** 0100 **Single Ended: ANA4, Differential: ANA4+, ANA5-** 0101 **Single Ended: ANA5, Differential: ANA4+, ANA5-** 0110 **Single Ended: ANA6, Differential: ANA6+, ANA7-** 0111 **Single Ended: ANA7, Differential: ANA6+, ANA7-** 1000 **Single Ended: ANB0, Differential: ANB0+, ANB1-** 1001 **Single Ended: ANB1, Differential: ANB0+, ANB1-** 1010 **Single Ended: ANB2, Differential: ANB2+, ANB3-** 1011 **Single Ended: ANB3, Differential: ANB2+, ANB3-** 1100 **Single Ended: ANB4, Differential: ANB4+, ANB5-** 1101 **Single Ended: ANB5, Differential: ANB4+, ANB5-** 1110 **Single Ended: ANB6, Differential: ANB6+, ANB7-** 1111 **Single Ended: ANB7, Differential: ANB6+, ANB7-**

25.3.9 ADC Sample Disable Register (ADCx_SDIS)

Address: Base address + 8h offset

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read Write	DS															
Reset	1	1	1	1	0	0	0	0	1	1	1	1	0	0	0	0

ADCx_SDIS field descriptions

Field	Description
DS	Disable Sample Bits 0 Enable CLIST*[SAMPLEx]. 1 Disable CLIST*[SAMPLEx] and all subsequent samples. Which samples are actually disabled will depend on the conversion mode, sequential/parallel, and the value of CTRL2[SIMULT].

25.3.10 ADC Status Register (ADCx_STAT)

This register provides the current status of the ADC module. STAT[HLMTI and LLMTI] bits are cleared by writing 1s to all asserted bits in the limit status register, LIMSTAT. Likewise, the STAT[ZCI] bit, is cleared by writing 1s to all asserted bits in the zero crossing status register, ZXSTAT. The STAT[EOSIx] bits are cleared by writing a one to them.

Except for STAT[CIP0 and CIP1] this register's bits are sticky. Once set to a one state, they require some specific action to clear them. They are not cleared automatically on the next scan sequence.

Address: Base address + 9h offset

Bit	15	14	13	12	11	10	9	8
Read	CIP0	CIP1	0	EOSI1	EOSI0	ZCI	LLMTI	HLMTI
Write								
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
Read	UNDEFINED							
Write								
Reset	0	0	0	0	0	0	0	0

ADCx_STAT field descriptions

Field	Description
15 CIP0	Conversion in Progress This bit indicates whether a scan is in progress. This refers to any scan except a B converter scan in non-simultaneous parallel scan modes. 0 Idle state 1 A scan cycle is in progress. The ADC will ignore all sync pulses or start commands
14 CIP1	Conversion in Progress This bit indicates whether a scan is in progress. This refers only to a B converter scan in non-simultaneous parallel scan modes. 0 Idle state 1 A scan cycle is in progress. The ADC will ignore all sync pulses or start commands
13 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
12 EOSI1	End of Scan Interrupt This bit indicates whether a scan of analog inputs have been completed since the last read of the status register or since a reset. This bit is cleared by writing a one to it. This bit cannot be set by software. In looping scan modes, this interrupt is triggered at the completion of each iteration of the loop.

Table continues on the next page...

ADCx_STAT field descriptions (continued)

Field	Description
	This interrupt is triggered only by the completion of a B converter scan in non-simultaneous parallel scan modes. 0 A scan cycle has not been completed, no end of scan IRQ pending 1 A scan cycle has been completed, end of scan IRQ pending
11 EOSI0	End of Scan Interrupt This bit indicates whether a scan of analog inputs have been completed since the last read of the status register or since a reset. This bit is cleared by writing a one to it. This bit cannot be set by software. STAT[EOSI0] is the preferred bit to poll for scan completion if interrupts are not enabled. In looping scan modes, this interrupt is triggered at the completion of each iteration of the loop. This interrupt is triggered upon the completion of any scan except for the completion of a B converter scan in non-simultaneous parallel scan modes. 0 A scan cycle has not been completed, no end of scan IRQ pending 1 A scan cycle has been completed, end of scan IRQ pending
10 ZCI	Zero Crossing Interrupt If the respective offset register is configured by having a value greater than 0000h, zero crossing checking is enabled. If the offset register is programmed with 7FF8h, the result will always be less than or equal to zero. On the other hand, if 0000h is programmed into the offset register, the result will always be greater than or equal to zero, and no zero crossing can occur because the sign of the result will not change. This interrupt asserts at the completion of an individual conversion which may or may not be the end of a scan. This bit is cleared by writing a "1" to all active ZXSTAT[ZCS] bits. 0 No zero crossing interrupt request 1 Zero crossing encountered, IRQ pending if CTRL1[ZCIE] is set
9 LLMTI	Low Limit Interrupt If the respective low limit register is enabled by having a value other than 0000h, low limit checking is enabled. This interrupt asserts at the completion of an individual conversion which may or may not be the end of a scan. This bit is cleared by writing a "1" to all active LIMSTAT[LLS] bits. 0 No low limit interrupt request 1 Low limit exceeded, IRQ pending if CTRL1[LLMTIE] is set
8 HLMTI	High Limit Interrupt If the respective high limit register is enabled by having a value other than 7FF8h, high limit checking is enabled. This interrupt asserts at the completion of an individual conversion which may or may not be the end of a scan. This bit is cleared by writing a "1" to all active LIMSTAT[HLS] bits. 0 No high limit interrupt request 1 High limit exceeded, IRQ pending if CTRL1[HLMTIE] is set
UNDEFINED	This read-only bitfield is undefined and will always contain random data.

25.3.11 ADC Ready Register (ADCx_RDY)

This register provides the current status of the ADC conversions. RDY[RDYx] bits are cleared by reading their corresponding result registers (RSLTx).

Address: Base address + Ah offset

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	RDY[15:0]															
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ADCx_RDY field descriptions

Field	Description
RDY[15:0]	Ready Sample These bits indicate samples fifteen through zero are ready to be read. These bits are cleared after a read from the respective results register. The RDY[RDYn] bits are set as the individual channel conversions are completed. Polling the RDY[RDYn] bits can determine if a particular sample is ready to be read. 0 Sample not ready or has been read 1 Sample ready to be read

25.3.12 ADC Low Limit Status Register (ADCx_LOLIMSTAT)

The low limit status register latches in the result of the comparison between the result of the sample and the respective low limit register, LOLIM0-15. Here is an example: If the result for the channel programmed in CLIST1[SAMPLE0] is lower than the value programmed into the Low Limit Register zero, then the LIMSTAT[LLS0] bit is set to one. An interrupt is generated if the CTRL1[LLMTIE] bit is set. These bits are sticky. They are not cleared automatically by subsequent conversions. Each bit is cleared only by writing a value of one to that specific bit.

Address: Base address + Bh offset

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	LLS[15:0]															
Write	w1c															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ADCx_LOLIMSTAT field descriptions

Field	Description
LLS[15:0]	Low Limit Status Bits

25.3.13 ADC High Limit Status Register (ADCx_HILIMSTAT)

The high limit status register latches in the result of the comparison between the result of the sample and the respective high limit register, HILIM0-15. Here is an example: If the result for the channel programmed in CLIST1[SAMPLE0] is greater than the value programmed into the High Limit Register zero, then the LIMSTAT[HLS0] bit is set to one. An interrupt is generated if the CTRL1[HLMTIE] bit is set. These bits are sticky. They are not cleared automatically by subsequent conversions. Each bit is cleared only by writing a value of one to that specific bit.

Address: Base address + Ch offset

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	HLS[15:0]															
Write	w1c															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ADCx_HILIMSTAT field descriptions

Field	Description
HLS[15:0]	High Limit Status Bits

25.3.14 ADC Zero Crossing Status Register (ADCx_ZXSTAT)

Address: Base address + Dh offset

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	ZCS[15:0]															
Write	w1c															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ADCx_ZXSTAT field descriptions

Field	Description
ZCS[15:0]	Zero Crossing Status The zero crossing condition is determined by examining the ADC value after it has been adjusted by the offset for the result register. Each bit of the register is cleared by writing a one to that register bit.

ADC*x*_ZXSTAT field descriptions (continued)

Field	Description
	0 Either: • A sign change did not occur in a comparison between the current channelx result and the previous channelx result, or • Zero crossing control is disabled for channelx in the zero crossing control register, ZXCTRL 1 In a comparison between the current channelx result and the previous channelx result, a sign change condition occurred as defined in the zero crossing control register (ZXCTRL)

25.3.15 ADC Result Registers with sign extension (ADC*x*_RSLT*n*)

The result registers contain the converted results from a scan. The CLIST1[SAMPLE0] result is loaded into RSLT0, CLIST1[SAMPLE1] result in RSLT1, and so on. In a parallel scan mode, the first channel pair designated by CLIST1[SAMPLE0] and CLIST3[SAMPLE8] are stored in RSLT0 and RSLT8, respectively.

Note

When writing to this register, only the RSLT portion of the value written is used. This value is modified and the result of the subtraction is stored. The SEXT bit is only set as a result of this subtraction and is not directly determined by the value written.

Address: Base address + Eh offset + (1d × i), where i=0d to 15d

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	SEXT	RSLT												0		
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ADC*x*_RSLT*n* field descriptions

Field	Description
15 SEXT	Sign Extend This is the sign-extend bit of the result. RSLT*[SEXT] set to one implies a negative result; RSLT*[SEXT] set to zero implies a positive result. If **unsigned** results are required, then the respective offset register must be set to a value of zero.
14–3 RSLT	Digital Result of the Conversion RSLT can be interpreted as either a signed integer or a signed fractional number. As a signed fractional number, the RSLT can be used directly. As a signed integer, it is an option to right shift with sign extend (ASR) three places and interpret the number, or accept the number as presented, knowing there are missing codes. The lower three bits are always going to be zero. Negative results, RSLT*[SEXT] = 1, are always presented in two's complement format. If it is a requirement of your application that the result registers always be positive, the offset registers must always be set to zero.

Table continues on the next page...

ADC*x*_RSLT*n* field descriptions (continued)

Field	Description
	The interpretation of the numbers programmed into the limit and offset registers, LOLIM, HILIM, and OFFST should match your interpretation of the result register.
Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

25.3.16 ADC Low Limit Registers (ADC*x*_LOLIM*n*)

Each ADC sample is compared against the values in the limit registers. The comparison is based upon the raw conversion value with no offset correction applied. The limit register used corresponds to the result register the value will be written to. The high limit register is used for the comparison of Result > High Limit. The low limit register is used for the comparison of Result < Low Limit. The limit checking can be disabled by programming the respective limit register with 7FF8h for the high limit and 0000h for the low limit. At reset, limit checking is disabled.

Address: Base address + 1Eh offset + (1d × i), where i=0d to 15d

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	0	LLMT												0		
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ADC*x*_LOLIM*n* field descriptions

Field	Description
15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
14–3 LLMT	Low Limit Bits
Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

25.3.17 ADC High Limit Registers (ADC*x*_HILIM*n*)

Each ADC sample is compared against the values in the limit registers. The comparison is based upon the raw conversion value with no offset correction applied. The limit register used corresponds to the result register the value will be written to. The high limit register is used for the comparison of Result > High Limit. The low limit register is used for the comparison of Result < Low Limit. The limit checking can be disabled by programming the respective limit register with 7FF8h for the high limit and 0000h for the low limit. At reset, limit checking is disabled.

Address: Base address + 2Eh offset + (1d × i), where i=0d to 15d

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	0	HLMT												0		
Write																
Reset	0	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0

ADC*x*_HILIM*n* field descriptions

Field	Description
15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
14–3 HLMT	High Limit Bits
Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

25.3.18 ADC Offset Registers (ADC*x*_OFFST*n*)

The value of the offset register is used to correct the ADC result before it is stored in the RSLT registers.

The offset value is subtracted from the ADC result. To obtain unsigned results, program the respective offset register with a value of $0000, thus giving a result range of $0000 to $7FF8.

Address: Base address + 3Eh offset + (1d × i), where i=0d to 15d

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	0	OFFSET												0		
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ADCx_OFFSTn field descriptions

Field	Description
15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
14–3 OFFSET	ADC Offset Bits
Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

25.3.19 ADC Power Control Register (ADCx_PWR)

This register controls the power management features of the ADC module. There are individual manual power down controls for the two ADC converters and the voltage reference generators. There are also five distinct power modes. The following terms are used to describe power modes and their related controls.

Power down state	Each converter and voltage reference generator can individually be put into a power down state. When powered down, the unit consumes no power. Results of scans referencing a powered down converter are undefined. At least one converter must be powered up to use the ADC module.
Manual power down controls	Each converter and voltage reference generator have a manual power control bit capable of putting that component into the power down state. Converters have other mechanisms that can automatically put them into the power down state.
Idle state	The ADC module is idle when neither of the two converters has a scan in process.
Active state	The ADC module is active when at least one of the two converters has a scan in process.
Current Mode	Both converters share a common current mode. Normal current mode is used to power the converters at clock rates above 600kHz. Current mode does not affect the number of ADC clock cycles required to do a conversion or the accuracy of a conversion. The ADC module may change the current mode when idle as part of the power saving strategy.
Startup delay	Auto-powerdown and auto-standby power modes cause a startup delay when the ADC module goes between the idle and active states to allow time to switch clocks or power configurations.

Address: Base address + 4Eh offset

Bit	15	14	13	12	11	10	9	8
Read	ASB	0		1	PSTS1	PSTS0	PUDELAY	
Write								
Reset	0	0	0	1	1	1	0	1

Bit	7	6	5	4	3	2	1	0
Read	PUDELAY				APD	1	PD1	PD0
Write								
Reset	1	0	1	0	0	1	1	1

ADCx_PWR field descriptions

Field	Description
15 ASB	Auto Standby This bit selects auto-standby mode. PWR[ASB] is ignored if PWR[APD] is 1. When the ADC is idle, auto-standby mode selects the standby clock as the ADC clock source and puts the converters into standby current mode. At the start of any scan, the conversion clock is selected as the ADC clock and then a delay of PWR[PUDELAY] ADC clock cycles is imposed for current levels to stabilize. After this delay, the ADC will initiate the scan. When the ADC returns to the idle state, the standby clock is again selected and the converters revert to the standby current state. This mode is not recommended for conversion clock rates at or below 100kHz. Instead, set PWR[ASB and APD]=0 and use standby power mode (normal mode with a sufficiently slow conversion clock so that standby current mode automatically engages). This provides the advantages of standby current mode while avoiding the clock switching and the PWR[PUDELAY]. Set PWR[ASB] prior to clearning PWR[PD1/0]. 0 Auto standby mode disabled 1 Auto standby mode enabled
14–13 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
12 Reserved	This field is reserved. This read-only field is reserved and always has the value 1.
11 PSTS1	ADC Converter B Power Status This bit is asserted immediately following a write of "1" to PWR[PD1]. It is de-asserted PWR[PUDELAY] ADC clock cycles after a write of "0" to PWR[PD1] if PWR[APD] is "0". This bit can be read as a status bit to determine when the ADC is ready for operation. During auto-powerdown mode, this bit indicates the current powered state of converter B. 0 ADC Converter B is currently powered up 1 ADC Converter B is currently powered down
10 PSTS0	ADC Converter A Power Status This bit is asserted immediately following a write of "1" to PWR[PD0]. It is de-asserted PWR[PUDELAY] ADC clock cycles after a write of "0" to PWR[PD0] if PWR[APD] is "0". This bit can be read as a status bit to determine when the ADC is ready for operation. During auto-powerdown mode, this bit indicates the current powered state of converter A. 0 ADC Converter A is currently powered up 1 ADC Converter A is currently powered down
9–4 PUDELAY	Power Up Delay This 6-bit field determines the number of ADC clocks provided to power up an ADC converter (after setting PWR[PD0 or PD1] to 0) before allowing a scan to start. It also determines the number of ADC clocks of delay provided in auto-powerdown (APD) and auto-standby (ASB) modes between when the ADC goes from the idle to active state and when the scan is allowed to start. The default value is 26 ADC clocks. Accuracy of the initial conversions in a scan will be degraded if PWR[PUDELAY] is set to too small a value.

Table continues on the next page...

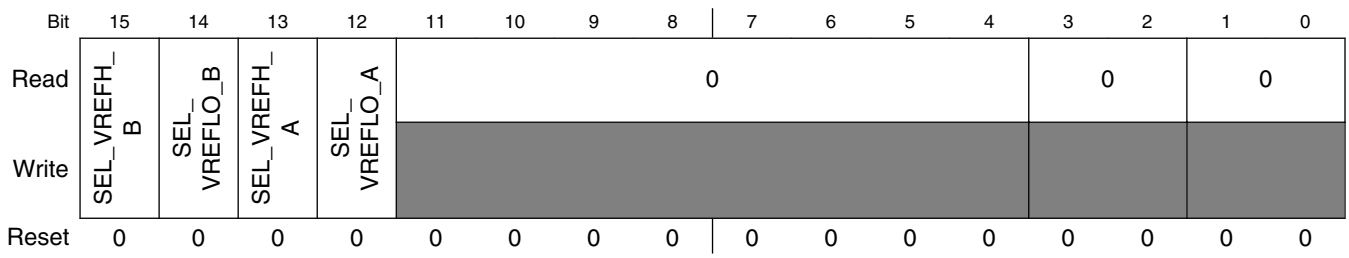
ADCx_PWR field descriptions (continued)

Field	Description
	NOTE: PWR[PUDELAY] defaults to a value that is typically sufficient for any power mode. The latency of a scan can be reduced by reducing PWR[PUDELAY] to the lowest value for which accuracy is not degraded. Refer to the data sheet for further details.
3 APD	Auto Powerdown Auto-powerdown mode powers down converters when not in use for a scan. PWR[APD] takes precedence over PWR[ASB]. When a scan is started in PWR[APD] mode, a delay of PWR[PUDELAY] ADC clock cycles is imposed during which the needed converter(s), if idle, are powered up. The ADC will then initiate a scan equivalent to that done when PWR[APD] is not active. When the scan is completed, the converter(s) are powered down again. **NOTE:** If PWR[ASB or APD] is asserted while a scan is in progress, that scan is unaffected and the ADC will wait to enter its low power state until after all conversions are complete and both ADCs are idle. PWR[ASB and APD] are not useful in looping modes. The continuous nature of scanning means that the low power state can never be entered. 0 Auto Powerdown Mode is not active 1 Auto Powerdown Mode is active
2 Reserved	This field is reserved. This read-only field is reserved and always has the value 1.
1 PD1	Manual Power Down for Converter B This bit forces ADC converter B to power down. Asserting this bit powers down converter B immediately. The results of a scan using converter B will be invalid while PWR[PD1] is asserted. When PWR[PD1] is cleared, converter B is either continuously powered up (PWR[APD] = 0) or automatically powered up when needed (PWR[APD]=1). When clearing this bit in any power mode except auto-powerdown (PWR[APD]=1), wait PWR[PUDELAY] ADC clock cycles before initiating a scan to stabilize power levels within the converter. The PWR[PSTS1] bit can be polled to determine when the PWR[PUDELAY] time has elapsed. Failure to follow this procedure can result in loss of accuracy of the first two samples. 0 Power Up ADC converter B 1 Power Down ADC converter B
0 PD0	Manual Power Down for Converter A This bit forces ADC converter A to power down. Asserting this bit powers down converter A immediately. The results of a scan using converter A will be invalid while PWR[PD0] is asserted. When PWR[PD0] is cleared, converter A is either continuously powered up (PWR[APD] = 0) or automatically powered up when needed (PWR[APD]=1). When clearing this bit in any power mode except auto-powerdown (PWR[APD]=1), wait PWR[PUDELAY] ADC clock cycles before initiating a scan to stabilize power levels within the converter. The PWR[PSTS0] bit can be polled to determine when the PWR[PUDELAY] time has elapsed. Failure to follow this procedure can result in loss of accuracy of the first two samples. 0 Power Up ADC converter A 1 Power Down ADC converter A

25.3.20 ADC Calibration Register (ADC*x*_CAL)

The ADC provides for off-chip references that can be used for ADC conversions.

Address: Base address + 4Fh offset



ADC*x*_CAL field descriptions

Field	Description
15 SEL_VREFH_B	Select V_{REFH} Source This bit selects the source of the V_{REFH} reference for all conversions in converter 1. 0 Internal VDDA 1 ANB2
14 SEL_VREFLO_B	Select V_{REFLO} Source This bit selects the source of the V_{REFLO} reference for all conversions in converter 1. 0 Internal VSSA 1 ANB3
13 SEL_VREFH_A	Select V_{REFH} Source This bit selects the source of the V_{REFH} reference for all conversions in converter 0. 0 Internal VDDA 1 ANA2
12 SEL_VREFLO_A	Select V_{REFLO} Source This bit selects the source of the V_{REFLO} reference for all conversions in converter 0. 0 Internal VSSA 1 ANA3
11–4 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
3–2 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

25.3.21 Gain Control 1 Register (ADCx_GC1)

The gain control registers are used to control amplification of each of the 16 input channels. GAIN0-GAIN7 control the amplification of inputs ANA0-ANA7 while GAIN8-GAIN15 control the amplification of inputs ANB0-ANB7.

Address: Base address + 50h offset

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read Write	GAIN7		GAIN6		GAIN5		GAIN4		GAIN3		GAIN2		GAIN1		GAIN0	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ADCx_GC1 field descriptions

Field	Description
15–14 GAIN7	Gain Control Bit 7 GAIN 7 controls ANA7 00 x1 amplification 01 x2 amplification 10 x4 amplification 11 reserved
13–12 GAIN6	Gain Control Bit 6 GAIN 6 controls ANA6 00 x1 amplification 01 x2 amplification 10 x4 amplification 11 reserved
11–10 GAIN5	Gain Control Bit 5 GAIN 5 controls ANA5 00 x1 amplification 01 x2 amplification 10 x4 amplification 11 reserved
9–8 GAIN4	Gain Control Bit 4 GAIN 4 controls ANA4 00 x1 amplification 01 x2 amplification 10 x4 amplification 11 reserved

Table continues on the next page...

ADCx_GC1 field descriptions (continued)

Field	Description
7–6 GAIN3	Gain Control Bit 3 GAIN 3 controls ANA3 00 x1 amplification 01 x2 amplification 10 x4 amplification 11 reserved
5–4 GAIN2	Gain Control Bit 2 GAIN 2 controls ANA2 00 x1 amplification 01 x2 amplification 10 x4 amplification 11 reserved
3–2 GAIN1	Gain Control Bit 1 GAIN 1 controls ANA1 00 x1 amplification 01 x2 amplification 10 x4 amplification 11 reserved
GAIN0	Gain Control Bit 0 GAIN 0 controls ANA0 00 x1 amplification 01 x2 amplification 10 x4 amplification 11 reserved

25.3.22 Gain Control 2 Register (ADCx_GC2)

The gain control registers are used to control amplification of each of the 16 input channels. GAIN0-GAIN7 control the amplification of inputs ANA0-ANA7 while GAIN8-GAIN15 control the amplification of inputs ANB0-ANB7.

Address: Base address + 51h offset

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read Write	GAIN15		GAIN14		GAIN13		GAIN12		GAIN11		GAIN10		GAIN9		GAIN8	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ADCx_GC2 field descriptions

Field	Description
15–14 GAIN15	Gain Control Bit 15 GAIN 15 controls ANB7 00 x1 amplification 01 x2 amplification 10 x4 amplification 11 reserved
13–12 GAIN14	Gain Control Bit 14 GAIN 14 controls ANB6 00 x1 amplification 01 x2 amplification 10 x4 amplification 11 reserved
11–10 GAIN13	Gain Control Bit 13 GAIN 13 controls ANB5 00 x1 amplification 01 x2 amplification 10 x4 amplification 11 reserved
9–8 GAIN12	Gain Control Bit 12 GAIN 12 controls ANB4 00 x1 amplification 01 x2 amplification 10 x4 amplification 11 reserved
7–6 GAIN11	Gain Control Bit 11 GAIN 11 controls ANB3 00 x1 amplification 01 x2 amplification 10 x4 amplification 11 reserved
5–4 GAIN10	Gain Control Bit 10 GAIN 10 controls ANB2 00 x1 amplification 01 x2 amplification 10 x4 amplification 11 reserved
3–2 GAIN9	Gain Control Bit 9 GAIN 9 controls ANB1

Table continues on the next page...

ADCx_GC2 field descriptions (continued)

Field	Description
	00 x1 amplification 01 x2 amplification 10 x4 amplification 11 reserved
GAIN8	Gain Control Bit 8 GAIN 8 controls ANB0 00 x1 amplification 01 x2 amplification 10 x4 amplification 11 reserved

25.3.23 ADC Scan Control Register (ADCx_SCTRL)

This register is an extension to the CLIST1-4 registers, providing the ability to pause and await a new sync while processing samples programmed in the CLIST*[SAMPLE0–SAMPLE15] fields.

These 16 control bits are used to determine whether a sample in a scan occurs immediately or if the sample waits for an enabled sync input to occur. The sync input must occur after the conversion of the current sample completes. During sequential mode scans, the SCTRL[SC] bits are used in order from SC0 to SC15. During simultaneous parallel scan modes, the bits are used in order from SC0 to SC3 and SC8-SC11. In non-simultaneous parallel scans, ADCA uses the bits in order from SC0 to SC3 followed by SC8 to SC11. ADCB will use bits SC4 to SC7 followed by SC12 to SC15 in non-simultaneous parallel scans.

When setting SCTRL[SC0], don't set CTRL1[START0] or CTRL2[START1]. Just clear CTRL1[STOP0] or CTRL2[STOP1] and the first enabled sync input will start the scan.

Setting SC0 delays sample 0 until a sync pulse occurs. Setting SC1 delays sample 1 until a sync pulse occurs after completing sample 0.

Address: Base address + 52h offset

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read Write	SC[15:0]															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ADCx_SCTRL field descriptions

Field	Description
SC[15:0]	Scan Control Bits

ADCx_SCTRL field descriptions (continued)

Field	Description
	0 Perform sample immediately after the completion of the current sample. 1 Delay sample until a new sync input occurs.

25.3.24 ADC Power Control Register (ADCx_PWR2)

Address: Base address + 53h offset

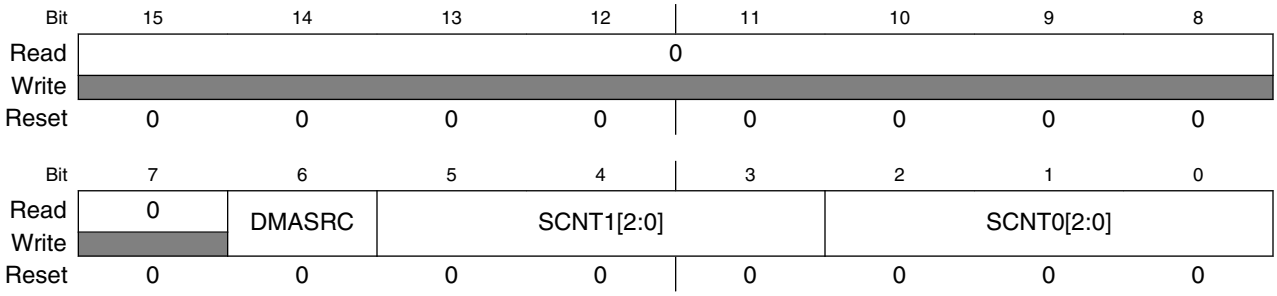
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	0		DIV1						0				SPEEDB		SPEEDA	
Write																
Reset	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0

ADCx_PWR2 field descriptions

Field	Description
15–14 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
13–8 DIV1	Clock Divisor Select The divider circuit operates in the same manner as the CTRL2[DIV0] field but is used to generate the clock used by ADCB during parallel non-simultaneous scan modes.
7–4 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
3–2 SPEEDB	ADCB Speed Control Bits These bits configure the clock speed at which the ADCB can operate. Faster conversion speeds require greater current consumption. 00 Conversion clock frequency ≤ 5 MHz; current consumption per converter = 5 mA 01 Conversion clock frequency ≤ 10 MHz; current consumption per converter = 9 mA 10 Conversion clock frequency ≤ 15 MHz; current consumption per converter = 15 mA 11 Conversion clock frequency ≤ 20 MHz; current consumption per converter = 21 mA
SPEEDA	ADCA Speed Control Bits These bits configure the clock speed at which the ADCA can operate. Faster conversion speeds require greater current consumption. 00 Conversion clock frequency ≤ 5 MHz; current consumption per converter = 5 mA 01 Conversion clock frequency ≤ 10 MHz; current consumption per converter = 9 mA 10 Conversion clock frequency ≤ 15 MHz; current consumption per converter = 15 mA 11 Conversion clock frequency ≤ 20 MHz; current consumption per converter = 21 mA

25.3.25 ADC Control Register 3 (ADCx_CTRL3)

Address: Base address + 54h offset



ADCx_CTRL3 field descriptions

Field	Description
15–7 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
6 DMASRC	DMA Trigger Source During sequential and simultaneous parallel scan modes CTRL3[DMASRC] selects between EOSI0 and RDY bits as the DMA source. During non-simultaneous parallel scan mode CTRL3[DMASRC] selects between EOSI0/EOSI1 for converters A and B, and the RDY bits as the DMA source. 0 DMA trigger source is end of scan interrupt 1 DMA trigger source is RDY bits
5–3 SCNT1[2:0]	Sample Window Count 1 During parallel non-simultaneous scan mode (CTRL2[SIMULT]=0) the CTRL3[SCNT1] bits are used to control the sampling time of the first sample after a scan is initiated on converter B. During sequential and parallel simultaneous scan modes, CTRL3[SCNT1] is ignored and the sampling window for converters A and B is controlled by CTRL3[SCNT0]. The default value is 0 which corresponds to a sampling time of 2 ADC clocks. Each increment of CTRL3[SCNT1] corresponds to an additional ADC clock cycle of sampling time with a maximum sampling time of 9 ADC clocks.
SCNT0[2:0]	Sample Window Count 0 During sequential and parallel simultaneous scan modes (CTRL2[SIMULT]=1) the CTRL3[SCNT0] bits control the sampling time of the first sample after a scan is initiated on both converters A and B. In parallel non-simultaneous mode (CTRL2[SIMULT]=0) CTRL3[SCNT0] affects converter A only. The default value is 0 which corresponds to a sampling time of 2 ADC clocks. Each increment of CTRL3[SCNT0] corresponds to an additional ADC clock cycle of sampling time with a maximum sampling time of 9 ADC clocks. In sequential scan mode the CTRL[SCNT0] setting will be ignored whenever the channel selected for the next sample is on the other converter. In other words, during a sequential scan, if a sample converts a converter A channel (ANA0-ANA7) and the next sample converts a converter B channel (ANB0-ANB7) or vice versa, CTRL3[SCNT0] will be ignored and use the default sampling time for the next sample.

25.3.26 ADC Scan Halted Interrupt Enable Register (ADCx_SCHLTEN)

This register is used with the scan control register (SCTRL) and the ready register (RDY) to select the samples that will generate a scan halted interrupt when the scan is paused by the SCTRL. Only the samples enabled in SCTRL should have their scan halted interrupt enables be set.

Address: Base address + 55h offset

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read Write	SCHLTEN[15:0]															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ADCx_SCHLTEN field descriptions

Field	Description
SCHLTEN[15:0]	Scan Halted Interrupt Enable 0 Scan halted interrupt is not enabled for this sample. 1 Scan halted interrupt is enabled for this sample.

25.4 Functional Description

The ADC consists of two eight-channel input select functions, which are two independent sample and hold (S/H) circuits feeding two separate 12-bit ADCs. The two separate converters store their results in an accessible buffer, awaiting further processing.

The conversion process is initiated either by a SYNC signal or by writing a 1 to a START bit.

Starting a single conversion actually begins a sequence of conversions, or a scan. The ADC operates in either sequential scan mode or parallel scan mode. In sequential scan mode, scan sequence is determined by defining sixteen sample slots that are processed in order, SAMPLE[0:15]. In parallel scan mode, converter A processes SAMPLE[0:7] in order, and converter B processes SAMPLE[8:15] in order. SAMPLE slots can be disabled using the ADC_SDIS control register to terminate a scan early.

In sequential scan mode, a scan takes up to sixteen single-ended or differential samples, one at a time. See the following figure.

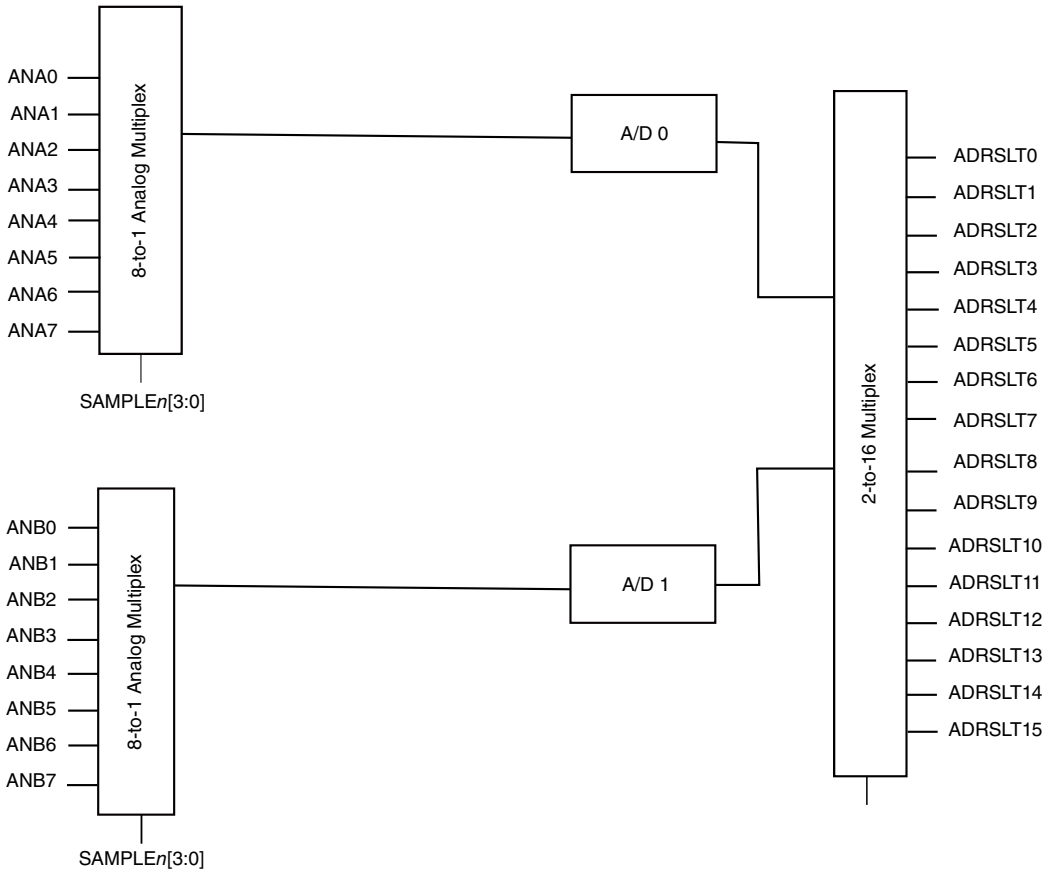


Figure 25-4. ADC Sequential Scan Mode

In parallel scan mode, eight of the sixteen samples are allocated to converter A and eight are allocated to converter B. Two converters operate in parallel, and each can take at most eight samples. Converter A can sample only analog inputs ANA[0:7], and converter B can sample only analog inputs ANB[0:7].

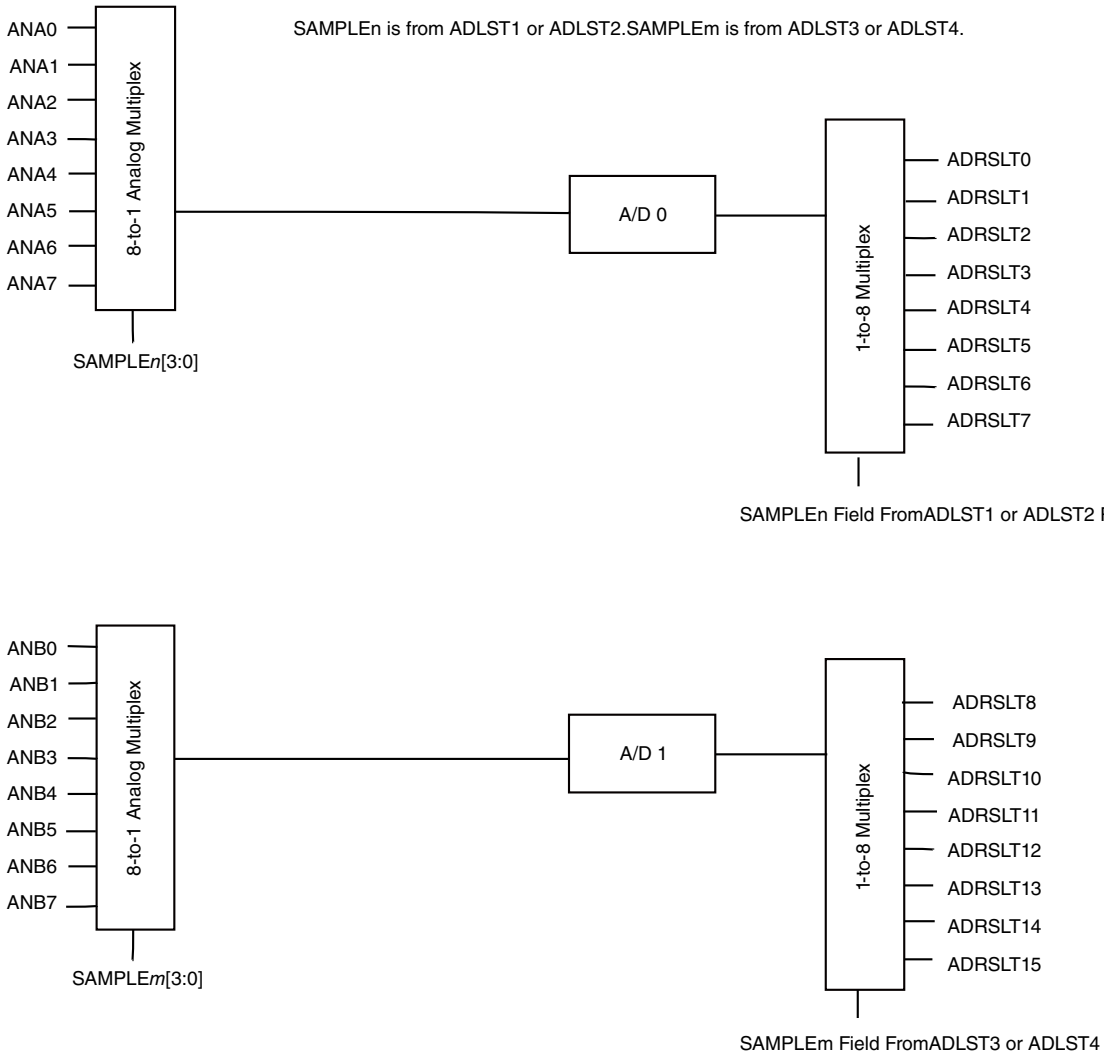


Figure 25-5. ADC Parallel Scan Mode

Each of the following pairs of analog inputs can be configured as a differential pair:

- ANA[0:1], ANA[2:3], ANA[4:5], ANA[6:7]
- ANB[0:1], ANB[2:3], ANB[4:5], ANB[6:7]

When they are so configured, a reference to either member of the differential pair by a sample slot results in a differential measurement using that differential pair.

Parallel scan mode can be simultaneous or non-simultaneous. In simultaneous scan mode, the parallel scans in the two converters occur simultaneously and result in simultaneous pairs of conversions, one by converter A and one by converter B. The two converters share the same start, stop, sync, end-of-scan interrupt enable control, and interrupts. Scanning in both converters terminates when either converter encounters a disabled sample. In non-simultaneous scan mode, the parallel scans in the two converters occur

independently. Each converter has its own start, stop, sync, end-of-scan interrupt enable controls, and interrupts. Scanning in either converter terminates only when that converter encounters a disabled sample.

The ADC can be configured to perform a single scan and halt, perform a scan whenever triggered, or perform the scan sequence repeatedly until manually stopped. The single scan (once mode) differs from the triggered mode only in that SYNC input signals must be re-armed after each use and subsequent SYNC inputs are ignored until the SYNC input is re-armed. This arming can occur any time after the SYNC pulse, including while the scan is still in process.

Optional interrupts can be generated at the end of a scan sequence. Interrupts are available simply to indicate that a scan has ended, that a sample is out of range, or several different zero crossing conditions. Range is determined by the high and low limit registers.

25.4.1 Input Multiplex Function

The following figure shows the input multiplex function. The ChannelSelect and SingleEndedDifferential switches are indirectly controlled by settings within the following registers:

- CLIST1, CLIST2, CLIST3, CLIST4, and SDIS registers
- CTRL1[CHNCFG_L]
- CTRL2[CHNCFG_H]

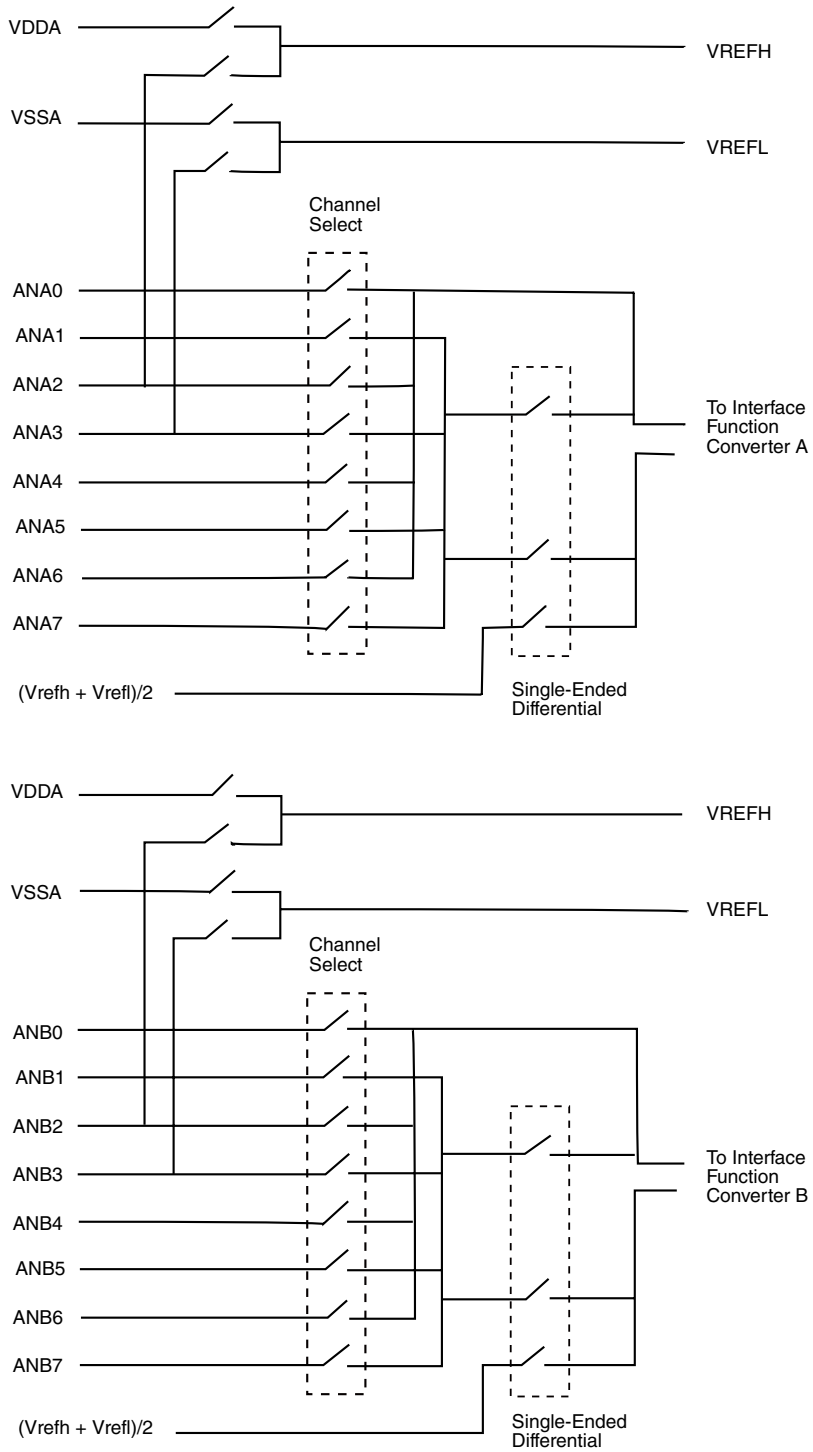


Figure 25-6. Input Select Multiplex

The multiplexing for conversions in different operating modes is as follows:

- Sequential, single-ended mode conversions — During each conversion cycle (sample), any one input of the two four input groups can be directed to its corresponding output.

- Sequential, differential mode conversions — During any conversion cycle (sample), either member of a differential pair may be referenced, resulting in a differential measurement on that pair.
- Parallel, single-ended mode conversions — During any conversion cycle (sample), any of ANA[0:7] can be directed to the converter A output and any of ANB[0:7] can be directed to the converter B output.
- Parallel, differential mode conversions — During any conversion cycle (sample), either member of any possible differential pair—ANA0/1, ANA2/3, ANA4/5, ANA6/7, ANB0/1, ANB2/3, ANB4/5, and ANB6/7—can be referenced, resulting in a differential measurement of that pair at the converter A output (for ANA pairs) or converter B output (for ANB pairs).

The details of single-ended and differential measurement are described under the CHNCFG field. Internally, all measurements are performed differentially.

Table 25-2. Analog Multiplexing Controls for Each Conversion Mode

Conversion Mode	Channel Select Switches	Single-Ended Differential Switches
General		The two lower switches within the dashed box are controlled so that one switch is always closed and the other open.
Sequential, single-ended	The two 1-of-8 select multiplexes can be set for the appropriate input line.	The lower switch is closed, providing $(V_{REFH}+V_{REFL})/2$ to the differential input of the A/D. The upper switch is always closed so that any of the four inputs can get to the A/D input.
Sequential, differential	The channel select switches are turned on in pairs, providing a dual 1-of-4 select function so that either of the two differential channels can be routed to the A/D input.	The upper and lower switches are open and the middle switch is closed, providing the differential channel to the differential input of the A/D.
Parallel, single-ended	The two 1-of-8 select multiplexes can be set for the appropriate input line.	The lower switch is closed, providing $V_{REF}/2$ to the differential input of the A/D. The upper switch is always closed so that any of the four inputs can get to the A/D input.
Parallel, differential	The channel select switches are turned on in pairs, providing a dual 1-of-4 select function so that either of the two differential channels can be routed to the A/D input.	The upper and lower switches are open and the middle switch is closed, providing the differential channel to the differential input of the A/D.

25.4.2 ADC Sample Conversion Operating Modes

The ADC consists of a cyclic, algorithmic architecture using two recursive sub-ranging sections (RSD 1 and RSD 2) as shown in the following figure. Each sub-ranging section resolves a single bit for each conversion clock, resulting in an overall conversion rate of 2 bits per clock cycle. Each sub-ranging section runs at a maximum clock speed of 20 MHz, so a complete 12-bit conversion can be accommodated in 300 ns, not including sample or post-processing time.

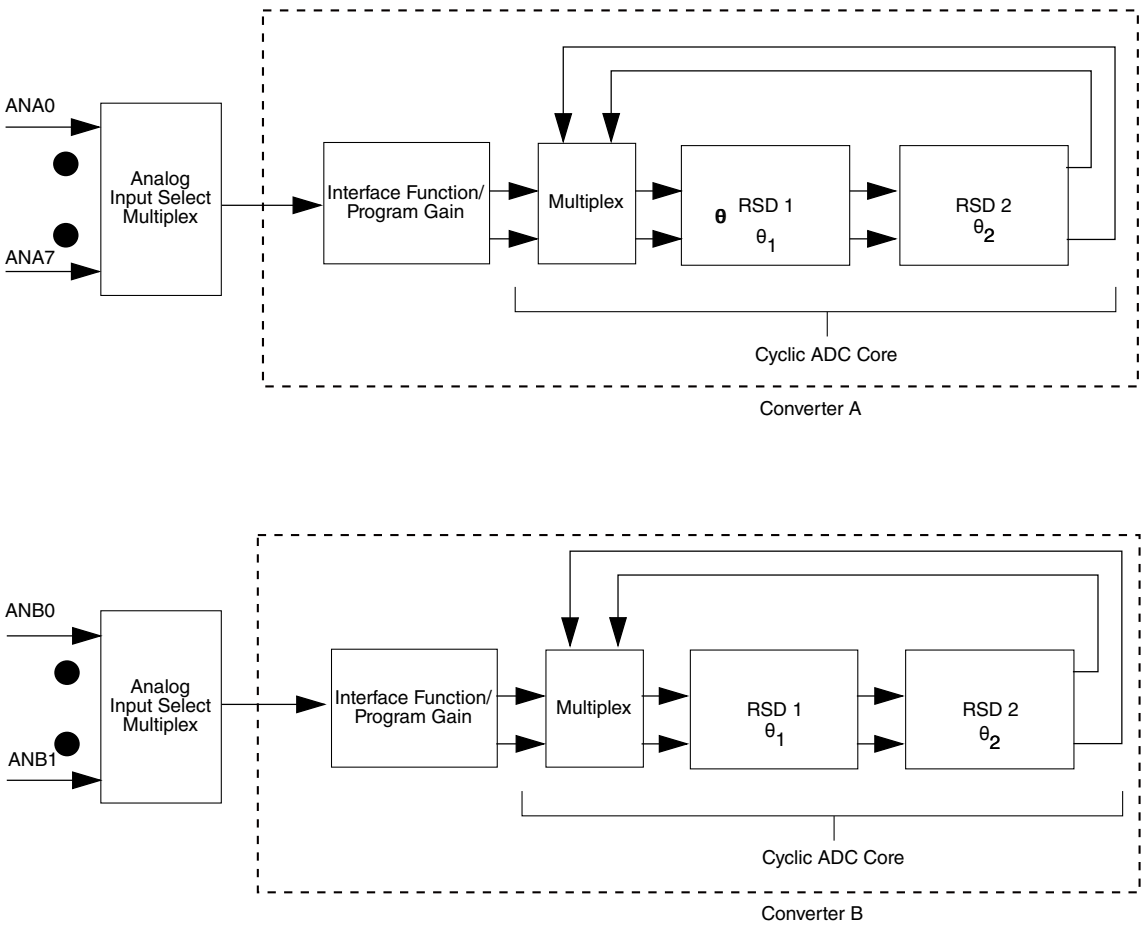


Figure 25-7. Top-Level Diagram of Cyclic ADC

25.4.2.1 Normal Mode Operation

The ADC has two normal operating modes: single-ended mode and differential mode. For a given sample, the mode of operation is determined by the CTRL1[CHNCFG] field:

- Single-ended mode (CHNCFG bit=0). The input multiplex of the ADC selects one of the 8 analog inputs and directs it to the plus terminal of the A/D core. The minus terminal of the A/D core is connected to the V_{REFL} reference. The ADC measures the voltage of the selected analog input and compares it against the (V_{REFH} - V_{REFL}) reference voltage range.
- Differential mode (CHNCFG bit=1). The ADC measures the voltage difference between two analog inputs and compares that value against the (V_{REFH}–V_{REFL}) voltage range. The input is selected as an input pair: ANA0/1, ANA2/3, ANA4/5, ANA6/7, ANB0/1, ANB2/3, ANB4/5, or ANB6/7. The plus terminal of the A/D core is connected to the even analog input, and the minus terminal is connected to the odd analog input.

A mix and match combination of single-ended and differential configurations may exist. For example:

- ANA[0:1] differential; ANA[2:3] single-ended
- ANA[4:5] differential; ANA[6:7] single-ended
- ANB[0:1] differential; ANB[2:3] single-ended
- ANB[4:5] differential; ANB[6:7] single-ended

25.4.2.1.1 Single-Ended Samples

The ADC module performs a ratio metric conversion. For single-ended measurements, the digital result is proportional to the ratio of the analog input to the reference voltage:

$$SingleEndedValue = round(((V_{IN} - V_{REFL})/(V_{REFH} - V_{REFL})) \times 4096) \times 8$$

V_{IN} is the applied voltage at the input pin.

V_{REFH} and V_{REFL} are the voltages at the external reference pins on the device (typically V_{REFH}=V_{DDA} and V_{REFL}=V_{SSA}).

Note: The 12-bit result is rounded to the nearest LSB.

Note: The ADC is a 12-bit function with 4096 possible states. However, the 12 bits have been left shifted by 3 bits on the 16-bit data bus. As a result, the magnitude of this function, as read from the data bus, is now 32760.

25.4.2.1.2 Differential Samples

For differential measurements, the digital result is proportional to the ratio of the difference in the inputs to the difference in the reference voltages (V_{REFH} and V_{REFL}).

When differential measurements are converted, the following formula is useful:

$$DifferentialValue = round\left(\left(\frac{VIN1-VIN2}{VREFH-VREFL} \times 2048\right) + 2048\right) \times 8$$

V_{IN}=Applied voltage at the input pin

V_{REFH} and V_{REFL}=Voltage at the external reference pins on the device (typically $V_{REFH}=V_{DDA}$ and $V_{REFL}=V_{SSA}$)

Note: The 12-bit result is rounded to the nearest LSB.

Note: The ADC is a 12-bit function with 4096 possible states. However, the 12 bits have been left shifted three bits on the 16-bit data bus, so the magnitude of this function, as read from the data bus, is now 32760.

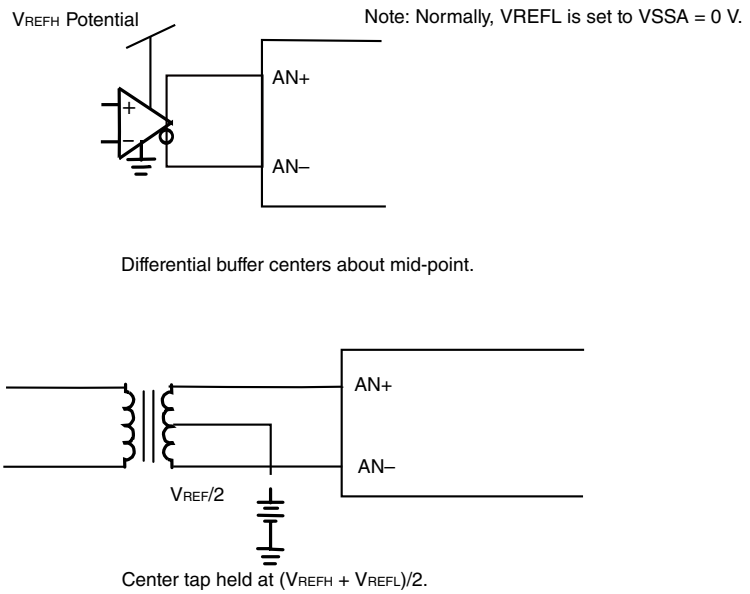


Figure 25-8. Typical Connections for Differential Measurements

25.4.3 ADC Data Processing

The result of an ADC conversion process is normally sent to an adder for offset correction, as the following figure shows. The adder subtracts the ADC_OFFST register value from each sample, and the resultant value is stored in the result register (RSLT). The raw ADC value and the RSLT values are checked for limit violations and zero-crossing, as shown. Appropriate interrupts are asserted, if enabled.

The result value sign is determined from the ADC unsigned result minus the respective offset register value. If the offset register is programmed with a value of zero, the result register value is unsigned and equals the cyclic converter unsigned result. The range of the result (RSLT) is 0000H–7FF8H, assuming that the offset register (OFFST) is cleared to all zeros. This is equal to the raw value of the ADC core.

The processor can write the result registers used for the results of a scan when the STOP bit for that scan is asserted. This write operation is treated as if it comes from the ADC analog core, so the limit checking, zero crossing, and the offset registers function as if in

normal mode. For example, if the STOP bit is set to one and the processor writes to RSLT5, the data written to the RSLT5 is multiplexed to the ADC digital logic inputs, processed, and stored into RSLT5 as if the analog core had provided the data. This test data must be justified, as illustrated by the RSLT register definition and does not include the sign bit.

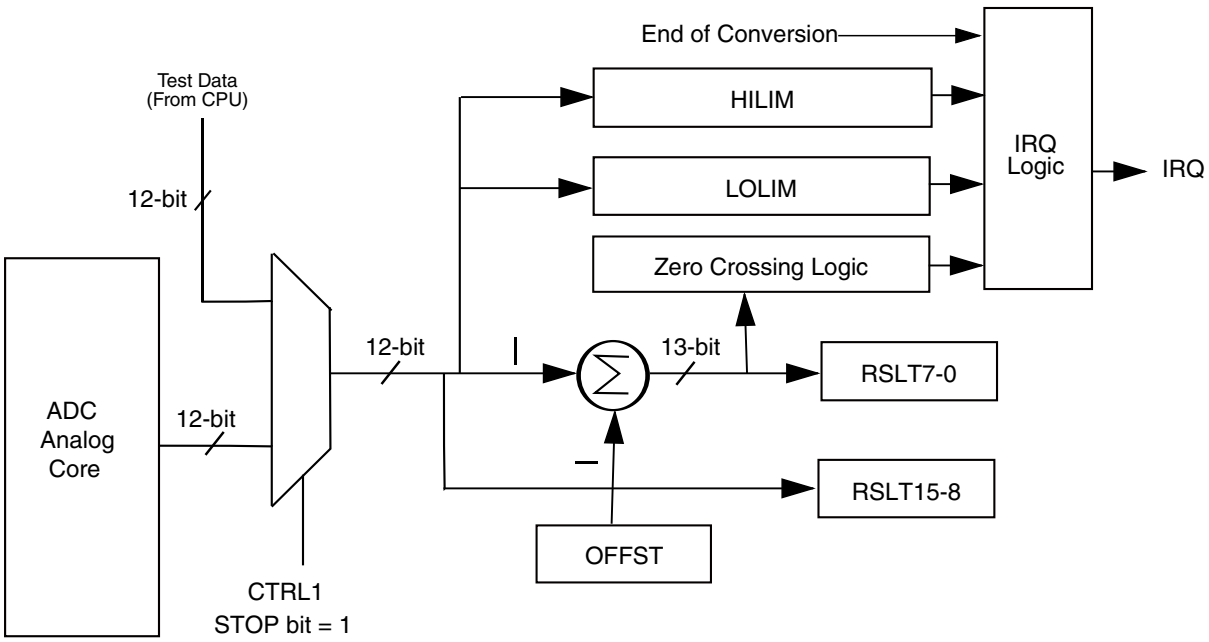


Figure 25-9. Result Register Data Manipulation

25.4.4 Sequential Versus Parallel Sampling

All scan modes use the sixteen sample slots in the CLIST1–4 registers. The slots are used to define which input or differential pair to measure at each step in a scan sequence. The SDIS register defines which sample slots are enabled. Input pairs ANA0/1, ANA2/3, ANA4/5, ANA6/7, ANB0/1, ANB2/3, ANB4/5, and ANB6/7 can be set to be measured differentially using the CHNCFG field. If a sample refers to an input that is not configured as a member of a differential pair, a single-ended measurement is made. If a sample refers to either member of a differential pair, a differential measurement is made.

Scan are either sequential or parallel. In sequential scans, up to sixteen sample slots are sampled one at a time in order, SAMPLE [0:15]. Each sample refers to any of the sixteen analog inputs ANA0–ANB7, so the same input can be referenced by more than one sample slot. All samples have the full functionality of offset subtraction and high/low limit compare. Scanning is initiated when the CTRL1 [START0] bit is written with a 1 or when the CTRL1[SYNC0] bit is set and the SYNC0 input goes high. A scan ends when the first disabled sample slot is encountered per the SDIS register. Completion of the scan triggers the STAT[EOSI0] interrupt if the CTRL1[EOSIEN0] interrupt enable is set. If

CTRL1[DMAEN0] is set and CTRL3[DMASRC]=0 a DMA transfer of the result data is initiated. The CTRL1[START0] bit and SYNC0 input are ignored while a scan is in process. Scanning stops and cannot be initiated when the CTRL1 [STOP0] bit is set.

Parallel scans differ in that converter A performs up to eight samples (SAMPLE[0:7]) in parallel with converter B (SAMPLE[8:15]). Constraints are as follows:

- SAMPLEs[0:7] can reference only the ANA[0:7] inputs.
- SAMPLEs[8:15] can reference only the ANB[0:7] inputs.

Within these constraints, any sample can reference any pin, and more than one sample slot can reference the same sample and the same input. All samples have the full functionality of offset subtraction and high/low limit compare. By default (when CTRL2[SIMULT]=1), the scans in both converters are initiated when the CTRL1[START0] bit is written with a 1 or when the CTRL1[SYNC0] bit has a value of 1 and the SYNC0 input goes high. The scan in both converters terminates when either converter encounters a disabled sample slot. Completion of a scan triggers the STAT[EOSI0] interrupt if the CTRL1[EOSIEN0] interrupt enable is set. If CTRL1[DMAEN0] is set and CTRL3[DMASRC]=0 then a DMA transfer of the result data is initiated. Samples are always taken simultaneously in both the A and B converters. Setting the CTRL1 [STOP0] bit stops and prevents the initiation of scanning in both converters.

Setting CTRL2[SIMULT]=0 (non-simultaneous mode) causes parallel scanning to operate independently in the A and B converter. Each converter has its own set of START, STOP, SYNC, DMAEN, and EOSIEN control bits, SYNC input, EOSI interrupt, and CIP status indicators (suffix 0 for converter A and suffix 1 for converter B). Though still operating in parallel, the scans in the A and B converter start and stop independently according to their own controls and can be simultaneous, phase shifted, or asynchronous depending on when scans are initiated on the respective converters. The A and B converters can be of different length (still up to a maximum of 8) and each converter's scan completes when a disabled sample is encountered in that converters sample list only. CTRL1[STOP0] stops the A converter only and CTRL2[STOP1] stops the B converter only. Looping scan modes iterate independently. Each converter independently restarts its scan after completing its list or encountering a disabled sample slot.

25.4.5 Scan Sequencing

The sequential and parallel scan modes fall into three types based on how they repeat:

- Once scan. A once scan executes a sequential or parallel scan only once each time it is started. It differs from a triggered scan in that sync inputs must be re-armed after each use.

- Triggered scan. Identical to the corresponding once scan modes except that resetting CTRL*[SYNC*] bits is not necessary.
- Looping scan. Automatically restarts a scan, either parallel or sequential, as soon as the previous scan completes. In parallel looping scan modes, the A converter scan restarts as soon as the A converter scan completes and the B converter scan restarts as soon as the B converter scan completes. All subsequent start and sync pulses are ignored after the scan begins unless the scan is paused by the SCTRL[SC] bits. Scanning can only be terminated by setting the STOP bit.

All scan modes ignore sync pulses while a scan is in process unless the scan is paused by the SCTRL[SC] bits. Once scan modes continue to ignore sync pulses even after the scan completes until the CTRL*[SYNC*] bit is set again. However, a reset can occur any time including during the scan. The SYNC0 input is re-armed by setting the CTRL1[SYNC0] bit, and the SYNC1 input is reset by setting the CTRL2[SYNC1] bit. A reset can be performed any time after a scan starts.

25.4.6 Enabling Additional Sample Slots for On-Chip Signals

The ADC supports up to four extra sample slots (2 for each converter) that are appended to the end of a scan on the ADC channels. These extra sample slots can only be assigned to on-chip generated analog signals which include the temperature sensor and the voltage regulator bandgap reference. None of the ADC channels can be assigned to these sample slots. The assignment to the extra sample slots is configured by ADC_CLIST5 and enabled using ADC_SDIS2. As conversion results for these samples become available the ADC_RDY2 status bits are asserted and the conversion data is stored in ADC_RSLT2*. Dedicated registers for ADC data processing (offset, zero crossing, limit checking) and ADC control (scan control, gain control) are supported for the extra sample slots.

NOTE

Enabling the four extra sample slots by themselves (ADC_SDIS=FFFF, and samples enabled in ADC_SDIS2) is supported only in once sequential mode. Sequential loop, and parallel (both sequential and loop) modes are not supported. It is suggested to poll the ADC_RDY2 register to check for conversion completion for this case.

25.4.7 Power Management

The five supported power modes are discussed in order from highest to lowest power usage at the expense of increased conversion latency and/or startup delay. Changes to the SIM and OCCS that affect the power modes should be made while the PWR[PD0] and PWR[PD1] bits are both asserted. See the Clocks section for details on the various clocks referenced here.

25.4.7.1 Low Power Modes

In the following table, the low-power modes are discussed in order from from highest to lowest power usage.

Mode	Description
Normal power	At least one ADC converter is powered up (PWR[PD0 or PD1] is 0), the PWR[APD and ASB] bits are both 0, and the SIM_PCE[ADC] bit is 1. The ADC uses the conversion clock as the ADC clock source in either active or idle. The conversion clock should be configured at or near 20 MHz to minimize conversion latency although PWR2[SPEEDn] can be used for reduced power consumption when lower conversion frequencies are acceptable . No startup delay (PWR[PUDELAY]) is imposed.
Auto-standby	At least one ADC converter is powered up (PWR[PD0 or PD1] is 0), PWR[APD] is 0, PWR[ASB] is 1, and the SIM_PCE[ADC] bit is 1. The OCCS MSTR_OSC signal must operate at 8 MHz. MSTR_OSC is divided by 80 to generate a 100 kHz standby clock either using an 8 MHz external clock source or using the ROSC as the clock source in its normal mode of operation (ROPD=ROSB=0). The ADC uses the conversion clock when active and the 100 kHz standby clock when idle. The standby (low current) state automatically engages when the ADC is idle. The conversion clock should be configured at or near 20 MHz to minimize conversion latency when active although PWR2[SPEEDn] can be used for reduced power consumption when lower conversion frequencies are acceptable. At the start of all scans, there is a startup delay of PWR[PUDELAY] ADC clocks to engage the conversion clock and revert from standby to normal current mode. Auto-standby is a compromise between normal and auto-powerdown modes. This mode offers moderate power savings at the cost of a moderate latency when leaving the idle state to start a new scan.
Auto-Powerdown	At least one ADC converter is powered up (PWR[PD0 or PD1] is 0), PWR[APD] is 1, and the SIM_PCE [ADC] bit is 1. The conversion clock should be configured at or near 20 MHz to minimize conversion latency when active although PWR2[SPEEDn] can be used for reduced power consumption when lower conversion frequencies are acceptable. The ADC uses the conversion clock when active. For maximum power savings, it gates off the conversion clock and powers down the converters when idle. At the start of all scans, there is a startup delay of PWR[PUDELAY] ADC clocks to stabilize normal current mode from a completely powered off condition. This mode saves more power than auto-standby but requires more startup latency when leaving the idle state to start a scan (higher PWR[PUDELAY] value).
Standby	At least one ADC converter is powered up (PWR[PD0 or PD1] is 0), the PWR[ASB] bit is 0, the SIM_PCE[ADC] bit is 1, the PLL is bypassed, and MSTR_OSC is driven from the ROSC in standby mode (PRECS=0, ROSB = 1 and ROPD = 0) at 400 kHz. The ADC clock operates continuously at 100 kHz and standby current mode is enabled continuously without loss of conversion accuracy. Though no startup delay (PWR[PUDELAY]) is imposed, the latency of a scan is affected by the low frequency of the ADC clock. Standby mode is not available when an external clock source is used because there is no way to determine that MSTR_OSC is driven at a low enough frequency to support this mode.

Table continues on the next page...

Mode	Description
	To use auto-powerdown mode and standby mode together, set PWR[APD]. This hybrid mode converts at an ADC clock rate of 100 kHz using standby current mode when active, and it gates off the ADC clock and powers down the converters when idle. At the start of all scans, there is a startup delay of PWR[PUDELAY] ADC clock cycles to engage the conversion clock and revert power-up the converters and stabilize them in the standby current mode. This is the slowest and lowest power operational configuration of the ADC.
Powerdown	Both ADC converters and voltage references are powered down (PWR[PD0 and PD1] are both 1) and the SIM_PCE[ADC] bit is 0. In this configuration, the clock trees to the ADC and all of its analog components are shut down and power utilization is eliminated.

25.4.7.2 Startup in Different Power Modes

The ADC voltage reference and converters are powered down (PWR[PDn]=1) on reset. Individual converters and voltage references can be manually powered down when not in use (PWR[PD0]=1 or PWR[PD1]=1). When the ADC reference is powered down, the output reference voltages are set to Low (VSSA) and the ADC data output is driven low.

A delay of PWR[PUDELAY] ADC clock cycles is imposed when PWR[PD0 or PD1] are cleared to power up a regulator and also to transition from an idle state in which neither converter has a scan in process to an active state in which at least one converter has a scan in process. ADC data sheets recommend the use of two PWR[PUDELAY] values: a large value for full powerup and a moderate value for transitioning from standby current levels to full powerup.

To start up in normal mode or standby power mode, perform the following steps:
1. Set PWR[PUDELAY] to the large power -up value.
2. Clear PWR[ASB and APD].
3. Clear the PWR[PD0 and/or PD1] bits to power up the required converters.
4. Poll the status bits until all required converters are powered up.
5. Start scan operations. This will provide a full power-up delay before scans begin.

Normal mode does not use PWR[PUDELAY] at start of scan, so no further delay is imposed.

To start up in auto-standby, use the normal mode startup procedure first. Before starting scan operations, set PWR[PUDELAY] to the moderate standby recovery value, and set PWR[ASB]. Auto- standby mode automatically reduces current levels until active and then imposes the PWR[PUDELAY] to allow current levels to rise from standby to full power levels.

To start up in auto powerdown mode, perform the following steps:
1. Set PWR[PUDELAY] to the large power-up value.
2. Clear PWR[ASB] and set PWR[APD].

3. Clear the PWR[PD0 and/or PD1] bits for the required converters.

Converters remain powered off until scanning goes active. Before a scan starts, there is a large PWR [PUDELAY] to go from the powered down to the fully powered state.

To avoid ambiguity and ensure that the proper delays are applied when powering up or starting scans, both regulators should be powered off (PWR[PD0]=PWR[PD1]=1) when the clock or power controls are configured.

Attempts to start a scan during the PWR[PUDELAY] are ignored until the appropriate PWR[PSTSn] bits are cleared.

Any attempt to use a converter when it is powered down or with the voltage references disabled will result in invalid results. It IS possible to read ADC result registers after converter power down for results calculated before power-down. A new scan sequence must be started with a SYNC pulse or a write to the START bit before new valid results are available.

In auto-powerdown mode, when the ADC goes from idle to active, a converter is powered up only if it is required for the scan as determined by the CLIST1-4 and SDIS registers.

25.4.7.3 Stop Mode of Operation

Any conversion sequence can be stopped by setting the relevant STOP bit. Any further sync pulses or writes to the start bit are ignored until the STOP bit is cleared. In stop mode, the results registers can be modified by writes from the processor. Any write to the result register in the ADC-STOP mode is treated as if the analog core supplied the data, so limit checking and zero crossing and associated interrupts can occur if enabled.

25.5 Reset

At reset, all the registers return to the reset state. The source of the single $\overline{\text{RST}}$ signal is the SIM.

25.6 Clocks

The ADC has two external clock inputs to drive two clock domains within the ADC module.

Table 25-3. Clock Summary

Clock input	Source	Characteristics
IP Clock	SIM	Maximum rate is 100 MHz. When the PLL is on and selected, it is PLL output divided by 4. When PLL is not selected, it is MSTR_OSC/2. When the device is in low-power mode, ROSB=1, the rate is 200 kHz.
adc_8_clk	ROSC clock	ROSC provides 8 MHz for auto-standby power saving mode.

The IP clock rate is determined by the OCCS module configuration, which is highly programmable. One of two sources (an external clock pin, or the ROSC) can be selected using the PRECS control to generate the IP_CLK. The maximum rate of the IP clock to the ADC is therefore either:

- 100 MHz based on PLL output with unity post-scaler divided by 5
- A maximum external clock rate of 200 MHz divided by 2

The IP_CLK is enabled only when the SIM_PCE[ADC] bit is set. This clock enable bit must be set before the ADC can be used.

The conversion clock is the primary source for the ADC clock and is always selected as the ADC clock when conversions are in process. The clock source controls in the OCCS (PRECS, ROPD, ROSB), and CTRL2[DIV0] and PWR2[DIV1] should be configured so that conversion clock frequency falls between 100 kHz and 20 MHz. Operating the ADC at out-of-spec conversion clock frequencies or reconfiguring the parameters that affect clock rates or power modes while the regulators are powered up (PWR[PD0]=0 or PWR[PD1]=0) negatively affects conversion accuracy.

The conversion clock that the ADC uses for sampling is calculated using the IP bus clock and the clock divisor bits within the ADC Control Register 2. The ADC clock is active 100 percent of the time in looping modes or in normal power mode. It is also active during all ADC powerup sequences for a period of time determined by the PWR[PUDELAY] field. If a conversion is initiated in power savings mode, then the ADC clock continues until the conversion sequence completes.

The following diagram shows the structure of the clocking system.

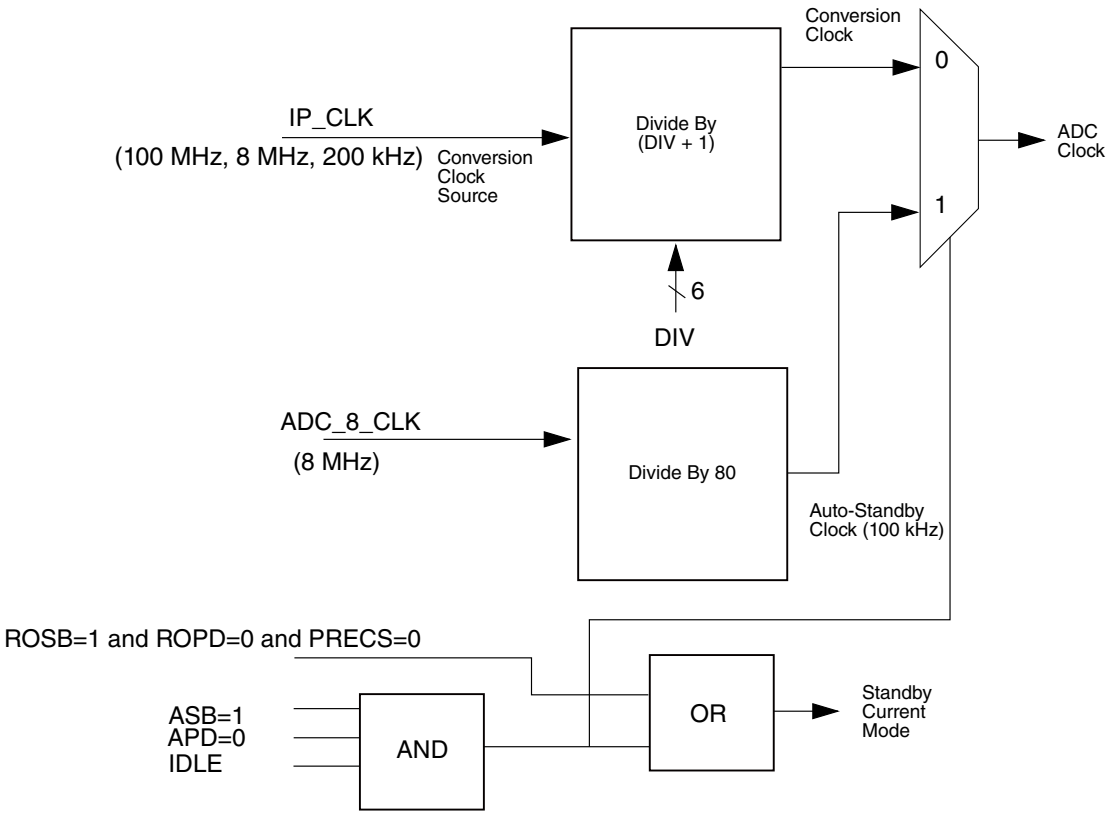


Figure 25-10. ADC Clock Generation

ADC_8_CLK clocks a divider to generate the auto-standby clock, which is selected as the ADC clock only during auto-standby power mode and only when both converters are idle. Auto-standby power mode requires the ADC_8_CLK to be 8 MHz using either the ROSC in normal mode or an 8 MHz external clock. The logic that selects the standby clock as the ADC clock also asserts standby current mode, which is available only at an ADC clock rate of less than 667 kHz. This mode provides substantially power savings yet requires less latency when switching back to the conversion clock at the start of a scan than auto-powerdown mode, which uses only the conversion clock as the ADC clock source but fully powers down the converters when idle.

The standby current mode is also engaged when IP_CLK is driven from the ROSC (PRECS=0) and the ROSC is in standby mode (ROSB=1 and ROPD=0). This ensures a 100 kHz ADC conversion clock rate while a conversion is in process. This configuration, referred to as standby power mode, provides accurate conversion without startup latency at the reduced power levels of standby current mode but requires a 100 kHz conversion clock. Standby power mode is not available with external clocking.

The ADC_CLK is an output of the gasket used to operate the two converters during scan operations. It is derived by multiplexing the conversion clock (divided version of the conversion clock source) and the standby clock (divided version of MSTR_OSC). This clock can be selected in the SIM for external output for debug and failure analysis.

25.7 Interrupts

The following table summarizes the ADC interrupts.

Table 25-4. Interrupt Summary

Interrupt	Source	Description
ADC_ERR_INT_B	STAT[[ZCI], STAT[LLMTI], STAT[HLMTI]	Zero Crossing, low Limit, and high limit interrupt
ADC_CC0_INT_B	STAT[EOSI0] RDY[RDY[15:0]]	Conversion Complete and Scan Halted Interrupt for any scan type except converter B scan in non-simultaneous parallel scan mode (see EOSI0)
ADC_CC1_INT_B	STAT[EOSI1] RDY[RDY[7:4]] RDY[RDY[15:12]]	Conversion Complete and Scan Halted Interrupt for converter B scan in non-simultaneous parallel scan mode (see EOSI1)

ADC interrupts fall into three categories:

- Threshold interrupts, which are caused by three different events. All of these interrupts are optional and enabled through control register CTRL1:
  - Zero crossing — occurs if the current result value has a sign change from the previous result as configured by the ZXCTRL register.
  - Low limit exceeded error — occurs when the current result value is less than the low limit register value. The raw result value is compared to LOLIM[LLMT] before the offset register value is subtracted.
  - High limit exceeded error — is asserted if the current result value is greater than the high limit register value. The raw result value is compared to HILIM[HLMT] before the offset register value is subtracted.
- Conversion complete interrupts, which are generated upon completion of any scan and convert sequence when CTRL1[EOSIE0]=1. Additional bits may need to be set in the Interrupt Control Module to enable the CPU to receive the interrupt signal.
- Scan halted interrupts, which are generated when a sample is converted and is paused by the SCTRL register. This allows processing of intermediate conversion data during a scan. The interrupt occurs when any sample has its SCTRL[SC] and SCHLTEN[SCHLTEN] bits enabled (set), and the RDY[RDY] bit for that sample is asserted. Use these registers to determine which sample triggered the interrupt.

25.8 Timing Specifications

The following figure shows a timing diagram for the ADC module. The ADC is assumed to be in Once or Triggered mode, so the ADC clock is shown in the OFF state prior to the SYNC pulse or START bit write. The ADC clock restarts (switching high) within 1 to 2 IP bus clocks of that event. ADC_CLK is derived from the ROSC or PLL output. The frequency relationship is programmable. Conversions are pipelined. The second start command is ignored because the ADC is busy with the previous start request. The third start command is recognized and is synchronized to the positive edge of the ADC clock when the conversion process is restarted. The ADC has two possible interrupts that are latched in the ADSTAT register:

- Conversion complete interrupt (End of Scan interrupt, EOSIx)
- Zero crossing or limit error interrupt (ZCI, LLMTI, and HLMTI)

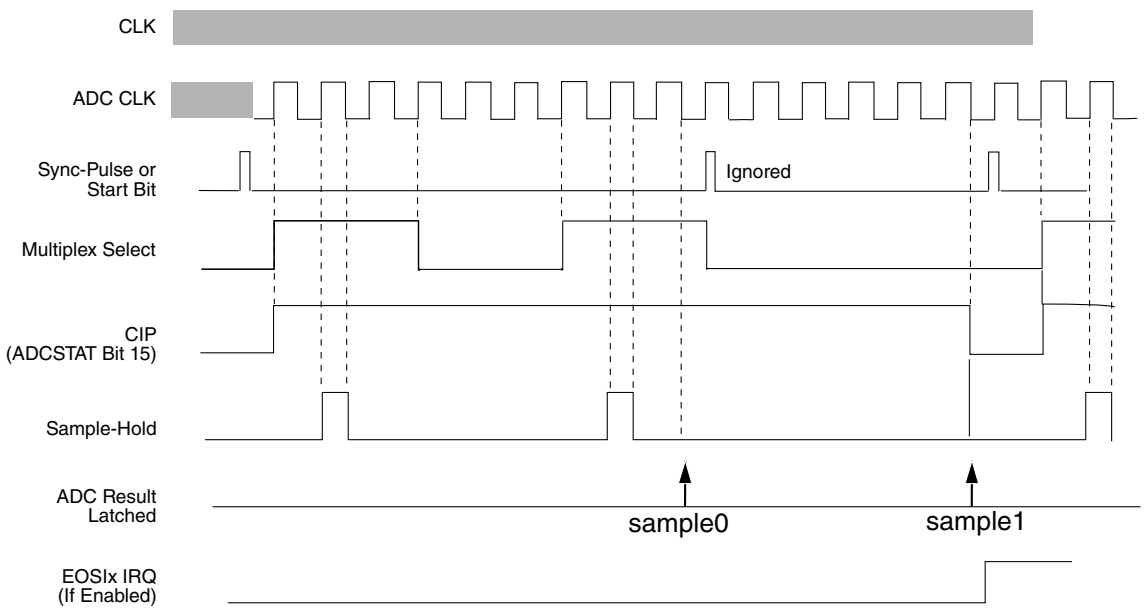


Figure 25-11. ADC Timing

As the figure shows, a conversion is initiated by (1) a sync pulse originating from the timer module or by (2) a write to a start bit. In APD or ASB mode, a delay of PWR [PUDELAY] ADC clock cycles is imposed. The conversion is initiated in the next clock cycle. The ADC clock period is determined by the CTRL2[DIV0] or PWR2[DIV1] value and the OCCS clock configuration.

The first conversion takes 8.5 ADC clocks to be valid. Then, each additional sample takes only six ADC clocks. The start conversion command is latched and the real conversion process is synchronized to the positive edge of the ADC clock.

Because the conversion is a pipeline process, after the last sample is in the S/H, the ADC cannot be restarted until the pipeline is emptied. However, the conversion cycle can be aborted by issuing a STOP command.

The figure shown here illustrates the case in which PWR[APD and ASB] are not in use. When the PWR[APD or ASB] bit is set, the sync pulse or start powers up the ADC, waits for a number of ADC clocks (determined by the PWR[PUDELAY] bits) for the ADC circuitry to stabilize, and only then begins the conversion sequence.

Chapter 26
Comparator (CMP)

26.1 Introduction

NOTE

For the chip-specific implementation details of this module's instances, see the chip configuration information.

The comparator (CMP) module provides a circuit for comparing two analog input voltages. The comparator circuit is designed to operate across the full range of the supply voltage, known as rail-to-rail operation.

The Analog MUX (ANMUX) provides a circuit for selecting an analog input signal from eight channels. One signal is provided by the 6-bit digital-to-analog converter (DAC). The mux circuit is designed to operate across the full range of the supply voltage.

The 6-bit DAC is 64-tap resistor ladder network which provides a selectable voltage reference for applications where voltage reference is needed. The 64-tap resistor ladder network divides the supply reference V_{in} into 64 voltage levels. A 6-bit digital signal input selects the output voltage level, which varies from V_{in} to $V_{in}/64$. V_{in} can be selected from two voltage sources, V_{in1} and V_{in2}. The 6-bit DAC from a comparator is available as an on-chip internal signal only and is not available externally to a pin.

26.1.1 CMP features

The CMP has the following features:

- Operational over the entire supply range
- Inputs may range from rail to rail
- Programmable hysteresis control

- Selectable interrupt on rising-edge, falling-edge, or both rising or falling edges of the comparator output
- Selectable inversion on comparator output
- Capability to produce a wide range of outputs such as:
  - Sampled
  - Windowed, which is ideal for certain PWM zero-crossing-detection applications
  - Digitally filtered:
    - Filter can be bypassed
    - Can be clocked via external SAMPLE signal or scaled bus clock
- External hysteresis can be used at the same time that the output filter is used for internal functions
- Two software selectable performance levels:
  - Shorter propagation delay at the expense of higher power
  - Low power, with longer propagation delay
- Functional in all modes of operation except VLLS0
- The window and filter functions are not available in the following modes:
  - Stop
  - VLPS
  - VLLSx

26.1.2 6-bit DAC key features

- 6-bit resolution
- Selectable supply reference source
- Power Down mode to conserve power when not in use
- Option to route the output to internal comparator input

26.1.3 ANMUX key features

- Two 8-to-1 channel mux
- Operational over the entire supply range

26.1.4 CMP, DAC and ANMUX diagram

The following figure shows the block diagram for the High-Speed Comparator, DAC, and ANMUX modules.

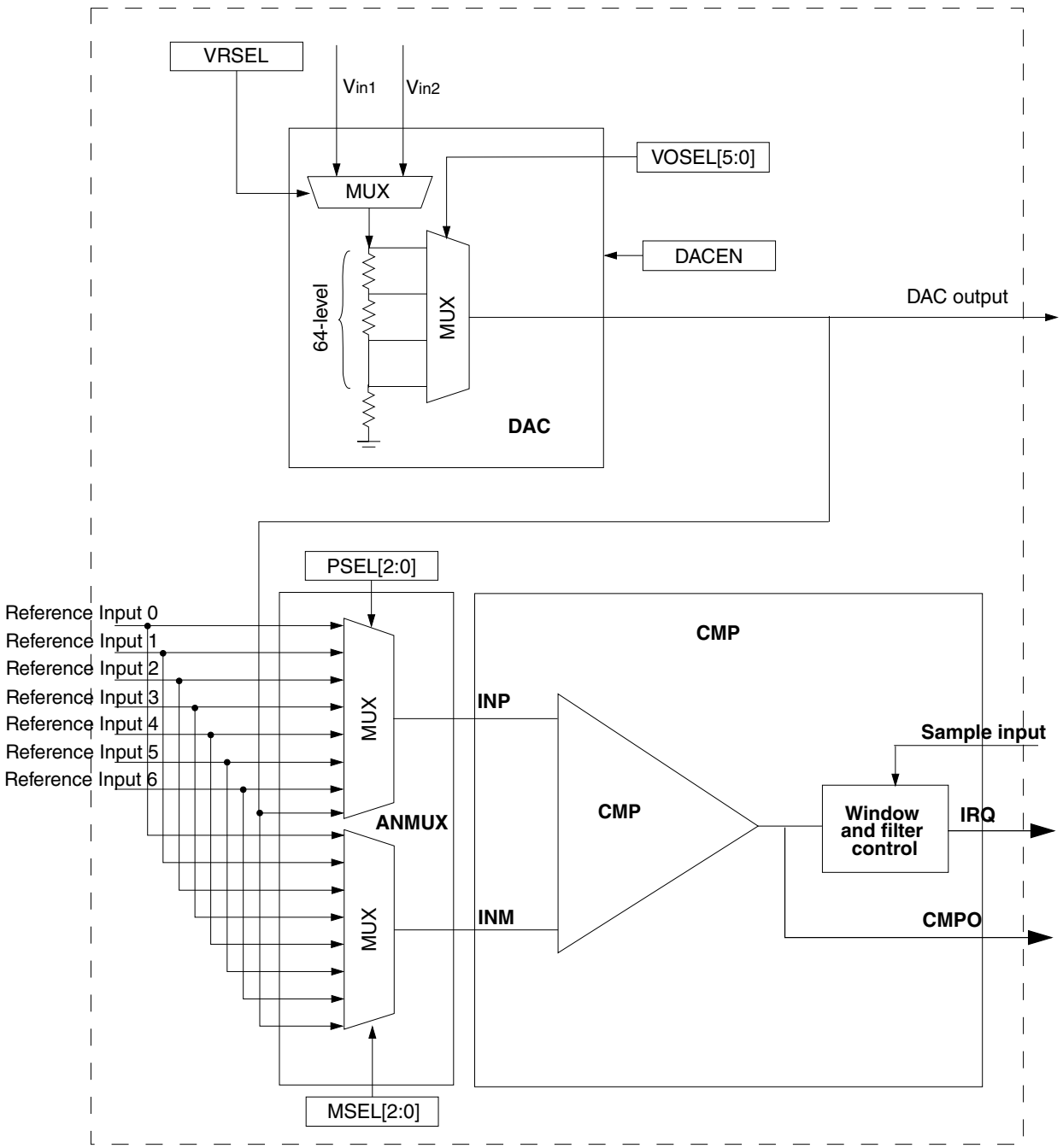


Figure 26-1. CMP, DAC and ANMUX block diagram

26.1.5 CMP block diagram

The following figure shows the block diagram for the CMP module.

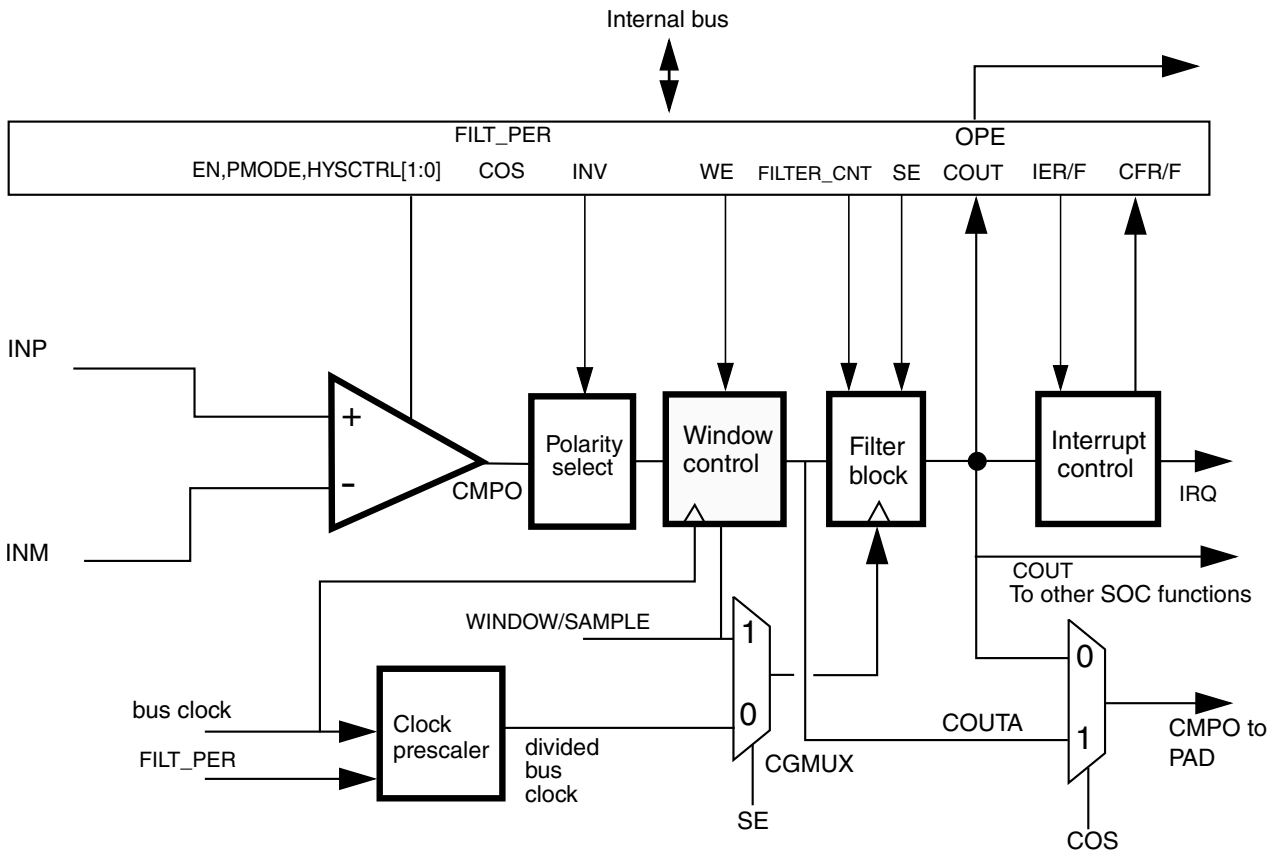


Figure 26-2. Comparator module block diagram

In the CMP block diagram:

- The Window Control block is bypassed when CR1[WE] = 0
- If CR1[WE] = 1, the comparator output will be sampled on every bus clock when WINDOW=1 to generate COUTA. Sampling does NOT occur when WINDOW = 0.
- The Filter block is bypassed when not in use.
- The Filter block acts as a simple sampler if the filter is bypassed and CR0[FILTER_CNT] is set to 0x01.
- The Filter block filters based on multiple samples when the filter is bypassed and CR0[FILTER_CNT] is set greater than 0x01.

- If CR1[SE] = 1, the external SAMPLE input is used as sampling clock
- If CR1[SE] = 0, the divided bus clock is used as sampling clock

- If enabled, the Filter block will incur up to one bus clock additional latency penalty on COUT due to the fact that COUT, which is crossing clock domain boundaries, must be resynchronized to the bus clock.
- CR1[WE] and CR1[SE] are mutually exclusive.

26.2 Memory map/register definitions

Address offsets are in terms of 16-bit words for core architectures. Each 8-bit register occupies bits 0-7 of the 16-bit width. The other 8 bits are read-only and always read 0.

CMP memory map

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
E020	CMP Control Register 0 (CMPA_CR0)	16	R/W	0000h	26.2.1/582
E021	CMP Control Register 1 (CMPA_CR1)	16	R/W	0000h	26.2.2/583
E022	CMP Filter Period Register (CMPA_FPR)	16	R/W	0000h	26.2.3/584
E023	CMP Status and Control Register (CMPA_SCR)	16	R/W	0000h	26.2.4/585
E024	DAC Control Register (CMPA_DACCR)	16	R/W	0000h	26.2.5/586
E025	MUX Control Register (CMPA_MUXCR)	16	R/W	0000h	26.2.6/587
E028	CMP Control Register 0 (CMPB_CR0)	16	R/W	0000h	26.2.1/582
E029	CMP Control Register 1 (CMPB_CR1)	16	R/W	0000h	26.2.2/583
E02A	CMP Filter Period Register (CMPB_FPR)	16	R/W	0000h	26.2.3/584
E02B	CMP Status and Control Register (CMPB_SCR)	16	R/W	0000h	26.2.4/585
E02C	DAC Control Register (CMPB_DACCR)	16	R/W	0000h	26.2.5/586
E02D	MUX Control Register (CMPB_MUXCR)	16	R/W	0000h	26.2.6/587
E030	CMP Control Register 0 (CMPC_CR0)	16	R/W	0000h	26.2.1/582
E031	CMP Control Register 1 (CMPC_CR1)	16	R/W	0000h	26.2.2/583
E032	CMP Filter Period Register (CMPC_FPR)	16	R/W	0000h	26.2.3/584
E033	CMP Status and Control Register (CMPC_SCR)	16	R/W	0000h	26.2.4/585
E034	DAC Control Register (CMPC_DACCR)	16	R/W	0000h	26.2.5/586
E035	MUX Control Register (CMPC_MUXCR)	16	R/W	0000h	26.2.6/587
E038	CMP Control Register 0 (CMPD_CR0)	16	R/W	0000h	26.2.1/582
E039	CMP Control Register 1 (CMPD_CR1)	16	R/W	0000h	26.2.2/583
E03A	CMP Filter Period Register (CMPD_FPR)	16	R/W	0000h	26.2.3/584

Table continues on the next page...

CMP memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
E03B	CMP Status and Control Register (CMPD_SCR)	16	R/W	0000h	26.2.4/585
E03C	DAC Control Register (CMPD_DACCR)	16	R/W	0000h	26.2.5/586
E03D	MUX Control Register (CMPD_MUXCR)	16	R/W	0000h	26.2.6/587

26.2.1 CMP Control Register 0 (CMPx_CR0)

Address: Base address + 0h offset

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	0								0	FILTER_CNT			0	0	HYSTCTR	
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

CMPx_CR0 field descriptions

Field	Description
15–8 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
7 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
6–4 FILTER_CNT	Filter Sample Count Represents the number of consecutive samples that must agree prior to the comparator ouput filter accepting a new output state. For information regarding filter programming and latency, see the Functional description. 000 Filter is disabled. If SE = 1, then COUT is a logic 0. This is not a legal state, and is not recommended. If SE = 0, COUT = COUTA. 001 One sample must agree. The comparator output is simply sampled. 010 2 consecutive samples must agree. 011 3 consecutive samples must agree. 100 4 consecutive samples must agree. 101 5 consecutive samples must agree. 110 6 consecutive samples must agree. 111 7 consecutive samples must agree.
3 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
2 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
HYSTCTR	Comparator hard block hysteresis control Defines the programmable hysteresis level. The hysteresis values associated with each level are device-specific. See the Data Sheet of the device for the exact values. 00 Level 0 01 Level 1

Table continues on the next page...

CMPx_CR0 field descriptions (continued)

Field	Description
	10 Level 2 11 Level 3

26.2.2 CMP Control Register 1 (CMPx_CR1)

Address: Base address + 1h offset

Bit	15	14	13	12	11	10	9	8
Read	0							
Write								
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
Read	SE	WE	0	PMODE	INV	COS	OPE	EN
Write								
Reset	0	0	0	0	0	0	0	0

CMPx_CR1 field descriptions

Field	Description
15–8 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
7 SE	Sample Enable At any given time, either SE or WE can be set. If a write to this register attempts to set both, then SE is set and WE is cleared. However, avoid writing 1s to both field locations because this "11" case is reserved and may change in future implementations. 0 Sampling mode is not selected. 1 Sampling mode is selected.
6 WE	Windowing Enable At any given time, either SE or WE can be set. If a write to this register attempts to set both, then SE is set and WE is cleared. However, avoid writing 1s to both field locations because this "11" case is reserved and may change in future implementations. 0 Windowing mode is not selected. 1 Windowing mode is selected.
5 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
4 PMODE	Power Mode Select See the electrical specifications table in the device Data Sheet for details. 0 Low-Speed (LS) Comparison mode selected. In this mode, CMP has slower output propagation delay and lower current consumption. 1 High-Speed (HS) Comparison mode selected. In this mode, CMP has faster output propagation delay and higher current consumption.

Table continues on the next page...

CMPx_CR1 field descriptions (continued)

Field	Description
3 INV	Comparator INVERT Allows selection of the polarity of the analog comparator function. It is also driven to the COUT output, on both the device pin and as SCR[COUT], when OPE=0. 0 Does not invert the comparator output. 1 Inverts the comparator output.
2 COS	Comparator Output Select 0 Set the filtered comparator output (CMPO) to equal COUT. 1 Set the unfiltered comparator output (CMPO) to equal COUTA.
1 OPE	Comparator Output Pin Enable 0 CMPO is not available on the associated CMPO output pin. If the comparator does not own the pin, this field has no effect. 1 CMPO is available on the associated CMPO output pin. The comparator output (CMPO) is driven out on the associated CMPO output pin if the comparator owns the pin. If the comparator does not own the field, this bit has no effect.
0 EN	Comparator Module Enable Enables the Analog Comparator module. When the module is not enabled, it remains in the off state, and consumes no power. When the user selects the same input from analog mux to the positive and negative port, the comparator is disabled automatically. 0 Analog Comparator is disabled. 1 Analog Comparator is enabled.

26.2.3 CMP Filter Period Register (CMPx_FPR)

Address: Base address + 2h offset

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	0								FILT_PER							
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

CMPx_FPR field descriptions

Field	Description
15–8 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
FILT_PER	Filter Sample Period Specifies the sampling period, in bus clock cycles, of the comparator output filter, when CR1[SE]=0. Setting FILT_PER to 0x0 disables the filter. Filter programming and latency details appear in the Functional description. This field has no effect when CR1[SE]=1. In that case, the external SAMPLE signal is used to determine the sampling period.

26.2.4 CMP Status and Control Register (CMPx_SCR)

Address: Base address + 3h offset

Bit	15	14	13	12	11	10	9	8
Read	0							
Write								
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
Read	0	Reserved	0	IER	IEF	CFR	CFF	COUT
Write						w1c	w1c	
Reset	0	0	0	0	0	0	0	0

CMPx_SCR field descriptions

Field	Description
15–8 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
7 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
6 Reserved	This field is reserved. This bit must be written as 0.
5 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
4 IER	Comparator Interrupt Enable Rising Enables the CFR interrupt from the CMP. When this field is set, an interrupt will be asserted when CFR is set. 0 Interrupt is disabled. 1 Interrupt is enabled.
3 IEF	Comparator Interrupt Enable Falling Enables the CFF interrupt from the CMP. When this field is set, an interrupt will be asserted when CFF is set. 0 Interrupt is disabled. 1 Interrupt is enabled.
2 CFR	Analog Comparator Flag Rising Detects a rising-edge on COUT, when set, during normal operation. CFR is cleared by writing 1 to it. During Stop modes, CFR . 0 Rising-edge on COUT has not been detected. 1 Rising-edge on COUT has occurred.
1 CFF	Analog Comparator Flag Falling Detects a falling-edge on COUT, when set, during normal operation. CFF is cleared by writing 1 to it. During Stop modes, CFF .

Table continues on the next page...

CMPx_SCR field descriptions (continued)

Field	Description
	0 Falling-edge on COUT has not been detected. 1 Falling-edge on COUT has occurred.
0 COUT	Analog Comparator Output Returns the current value of the Analog Comparator output, when read. The field is reset to 0 and will read as CR1[INV] when the Analog Comparator module is disabled, that is, when CR1[EN] = 0. Writes to this field are ignored.

26.2.5 DAC Control Register (CMPx_DACCR)

Address: Base address + 4h offset

Bit	15	14	13	12	11	10	9	8
Read	0							
Write								
Reset	0	0	0	0	0	0	0	0

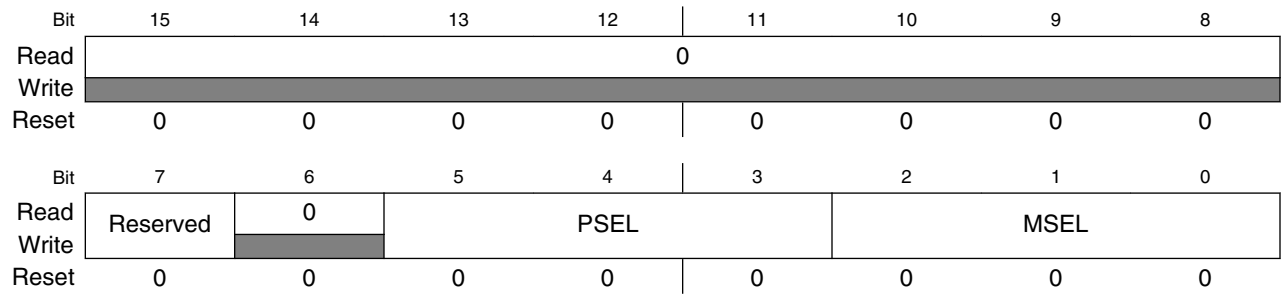
Bit	7	6	5	4	3	2	1	0
Read Write	DACEN	VRSEL	VOSEL					
Reset	0	0	0	0	0	0	0	0

CMPx_DACCR field descriptions

Field	Description
15–8 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
7 DACEN	DAC Enable Enables the DAC. When the DAC is disabled, it is powered down to conserve power. 0 DAC is disabled. 1 DAC is enabled.
6 VRSEL	Supply Voltage Reference Source Select 0 V_{in1} is selected as resistor ladder network supply reference. 1 V_{in2} is selected as resistor ladder network supply reference.
VOSEL	DAC Output Voltage Select Selects an output voltage from one of 64 distinct levels. `DACO = (V in /64) * (VOSEL[5:0] + 1)` , so the DACO range is from V_{in} /64 to V_{in} .

26.2.6 MUX Control Register (CMPx_MUXCR)

Address: Base address + 5h offset



CMPx_MUXCR field descriptions

Field	Description
15–8 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
7 Reserved	Bit can be programmed to zero only . This field is reserved.
6 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
5–3 PSEL	Plus Input Mux Control Determines which input is selected for the plus input of the comparator. For INx inputs, see CMP, DAC, and ANMUX block diagrams. **NOTE:** When an inappropriate operation selects the same input for both muxes, the comparator automatically shuts down to prevent itself from becoming a noise generator. 000 IN0 001 IN1 010 IN2 011 IN3 100 IN4 101 IN5 110 IN6 111 IN7
MSEL	Minus Input Mux Control Determines which input is selected for the minus input of the comparator. For INx inputs, see CMP, DAC, and ANMUX block diagrams. **NOTE:** When an inappropriate operation selects the same input for both muxes, the comparator automatically shuts down to prevent itself from becoming a noise generator. 000 IN0 001 IN1 010 IN2 011 IN3 100 IN4

Table continues on the next page...

CMPx_MUXCR field descriptions (continued)

Field	Description
	101 IN5 110 IN6 111 IN7

26.3 Functional description

The CMP module can be used to compare two analog input voltages applied to INP and INM.

CMPO is high when the non-inverting input is greater than the inverting input, and is low when the non-inverting input is less than the inverting input. This signal can be selectively inverted by setting CR1[INV] = 1.

SCR[IER] and SCR[IEF] are used to select the condition which will cause the CMP module to assert an interrupt to the processor. SCR[CFF] is set on a falling-edge and SCR[CFR] is set on rising-edge of the comparator output. The optionally filtered CMPO can be read directly through SCR[COUT].

26.3.1 CMP functional modes

There are the following main sub-blocks to the CMP module:

- The comparator itself
- The window function
- The filter function

The filter, CR0[FILTER_CNT], can be clocked from an internal or external clock source. The filter is programmable with respect to the number of samples that must agree before a change in the output is registered. In the simplest case, only one sample must agree. In this case, the filter acts as a simple sampler.

The external sample input is enabled using CR1[SE]. When set, the output of the comparator is sampled only on rising edges of the sample input.

The "windowing mode" is enabled by setting CR1[WE]. When set, the comparator output is sampled only when WINDOW=1. This feature can be used to ignore the comparator output during time periods in which the input voltages are not valid. This is especially useful when implementing zero-crossing-detection for certain PWM applications.

The comparator filter and sampling features can be combined as shown in the following table. Individual modes are discussed below.

Table 26-1. Comparator sample/filter controls

Mode #	CR1[EN]	CR1[WE]	CR1[SE]	CR0[FILTER_CNT]	FPR[FILT_PER]	Operation
1	0	X	X	X	X	**Disabled** See the Disabled mode (# 1).
2A	1	0	0	0x00	X	**Continuous Mode** See the Continuous mode (#s 2A & 2B).
2B	1	0	0	X	0x00	
3A	1	0	1	0x01	X	**Sampled, Non-Filtered mode** See the Sampled, Non-Filtered mode (#s 3A & 3B).
3B	1	0	0	0x01	> 0x00	
4A	1	0	1	> 0x01	X	**Sampled, Filtered mode** See the Sampled, Filtered mode (#s 4A & 4B).
4B	1	0	0	> 0x01	> 0x00	
5A	1	1	0	0x00	X	**Windowed mode** Comparator output is sampled on every rising bus clock edge when SAMPLE=1 to generate COUTA. See the Windowed mode (#s 5A & 5B).
5B	1	1	0	X	0x00	
6	1	1	0	0x01	0x01–0xFF	**Windowed/Resampled mode** Comparator output is sampled on every rising bus clock edge when SAMPLE=1 to generate COUTA, which is then resampled on an interval determined by FILT_PER to generate COUT. See the Windowed/Resampled mode (# 6).
7	1	1	0	> 0x01	0x01–0xFF	**Windowed/Filtered mode** Comparator output is sampled on every rising bus clock edge when SAMPLE=1 to generate COUTA, which is then resampled and filtered to generate COUT. See the Windowed/Filtered mode (#7).
All other combinations of CR1[EN], CR1[WE], CR1[SE], CR0[FILTER_CNT], and FPR[FILT_PER] are illegal.						

For cases where a comparator is used to drive a fault input, for example, for a inverter-control module such as PWM, it must be configured to operate in Continuous mode so that an external fault can immediately pass through the comparator to the target fault circuitry.

Note

Filtering and sampling settings must be changed only after setting CR1[SE]=0 and CR0[FILTER_CNT]=0x00. This resets the filter to a known state.

26.3.1.1 Disabled mode (# 1)

In Disabled mode, the analog comparator is non-functional and consumes no power. CMPO is 0 in this mode.

26.3.1.2 Continuous mode (#s 2A & 2B)

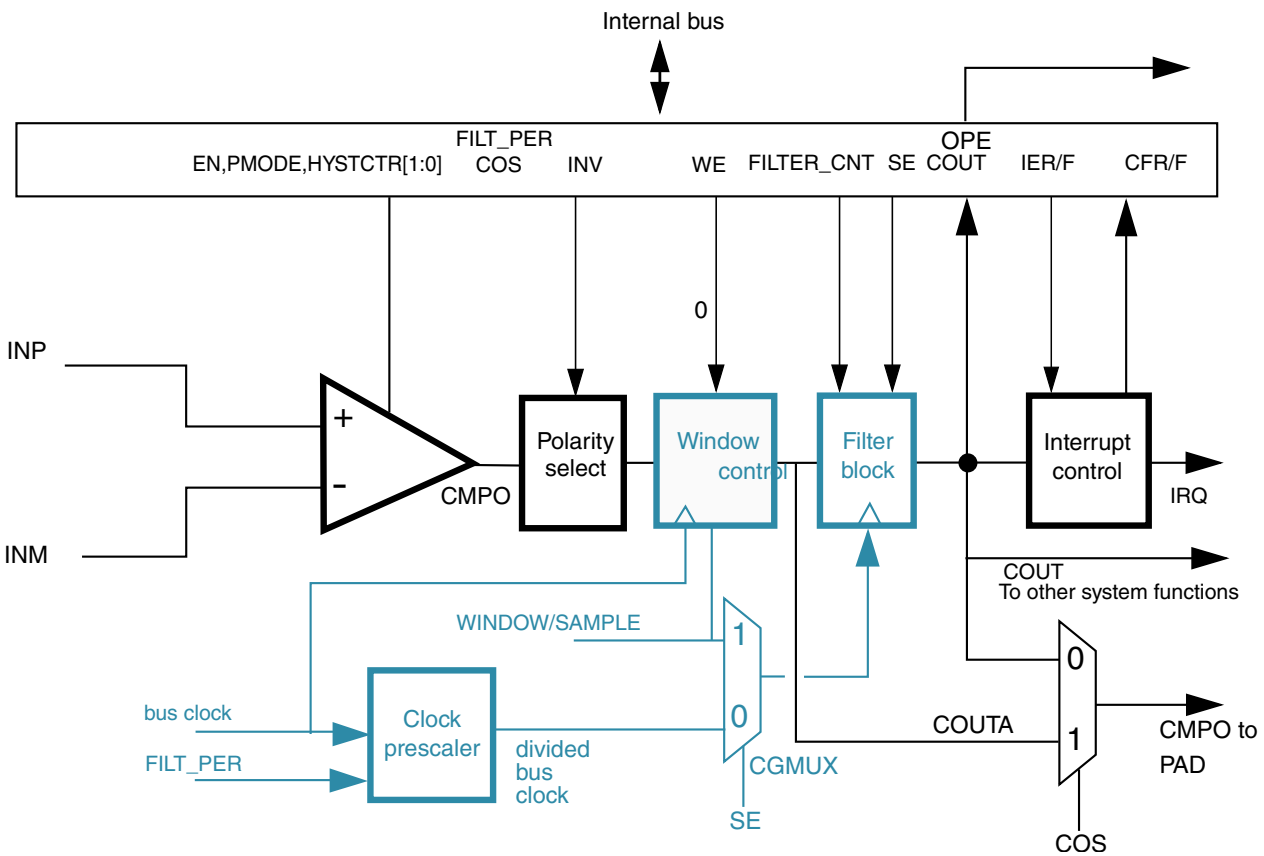


Figure 26-3. Comparator operation in Continuous mode

NOTE

See the chip configuration section for the source of sample/ window input.

The analog comparator block is powered and active. CMPO may be optionally inverted, but is not subject to external sampling or filtering. Both window control and filter blocks are completely bypassed. SCR[COUT] is updated continuously. The path from comparator input pins to output pin is operating in combinational unclocked mode. COUT and COUTA are identical.

For control configurations which result in disabling the filter block, see the Filter Block Bypass Logic diagram.

26.3.1.3 Sampled, Non-Filtered mode (#s 3A & 3B)

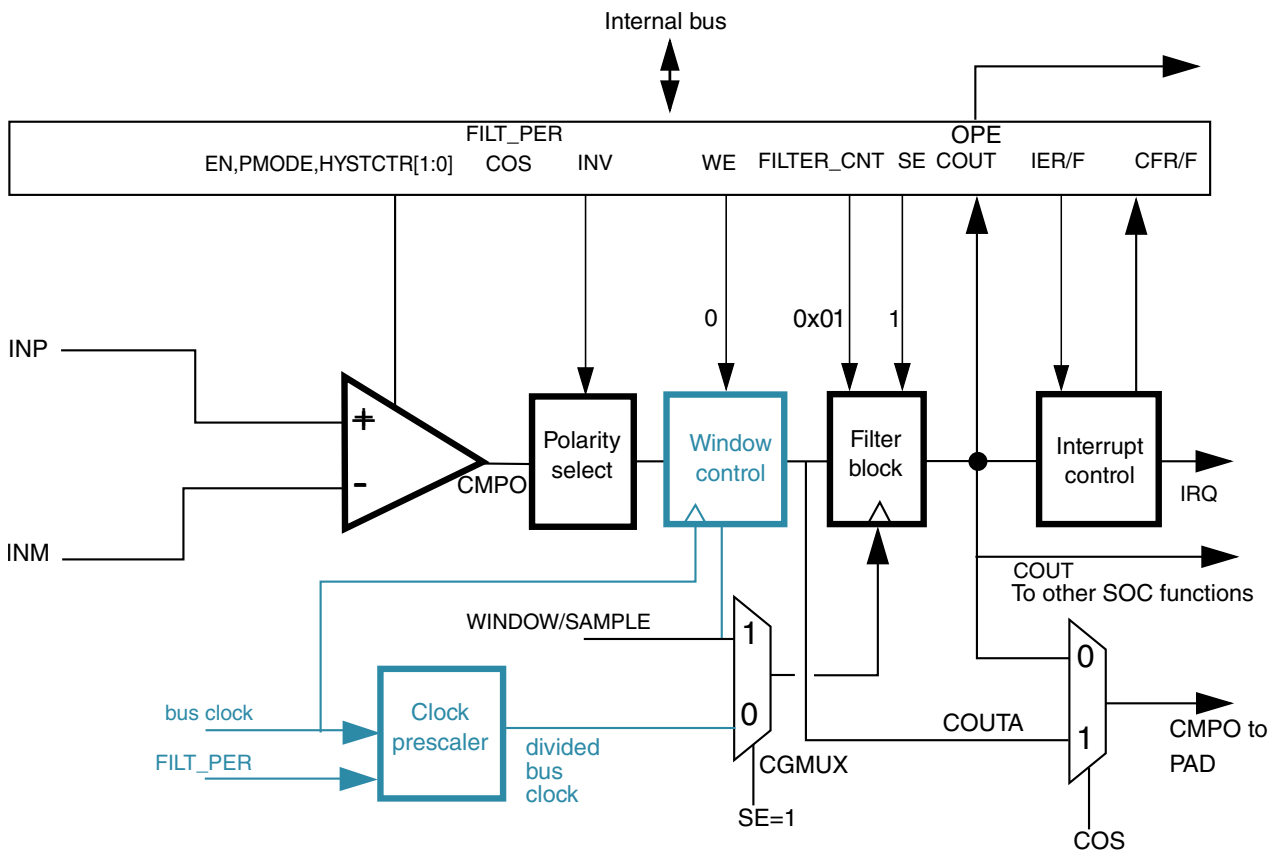


Figure 26-4. Sampled, Non-Filtered (# 3A): sampling point externally driven

In Sampled, Non-Filtered mode, the analog comparator block is powered and active. The path from analog inputs to COUTA is combinational unclocked. Windowing control is completely bypassed. COUTA is sampled whenever a rising-edge is detected on the filter block clock input.

The only difference in operation between Sampled, Non-Filtered (# 3A) and Sampled, Non-Filtered (# 3B) is in how the clock to the filter block is derived. In #3A, the clock to filter block is externally derived while in #3B, the clock to filter block is internally derived.

The comparator filter has no other function than sample/hold of the comparator output in this mode (# 3B).

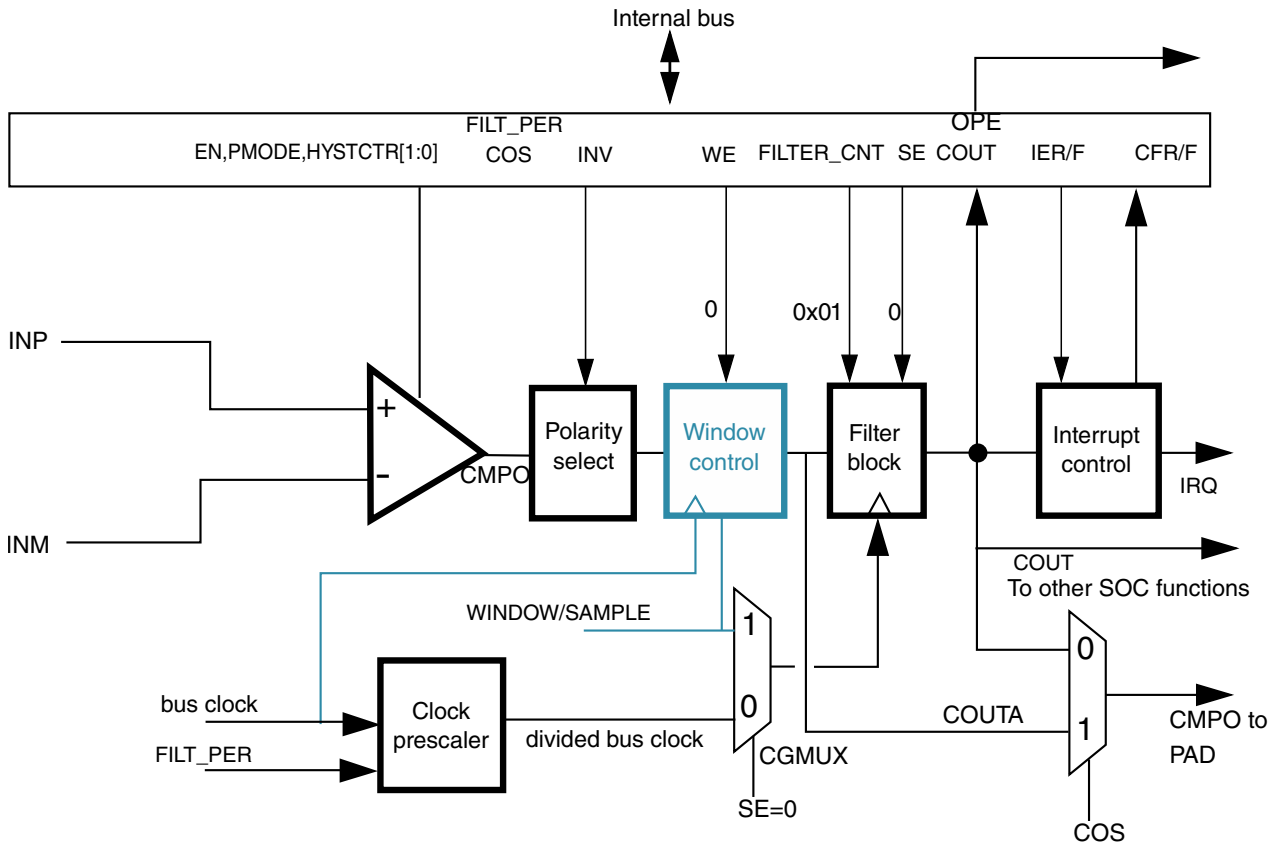


Figure 26-5. Sampled, Non-Filtered (# 3B): sampling interval internally derived

26.3.1.4 Sampled, Filtered mode (#s 4A & 4B)

In Sampled, Filtered mode, the analog comparator block is powered and active. The path from analog inputs to COUTA is combinational unclocked. Windowing control is completely bypassed. COUTA is sampled whenever a rising edge is detected on the filter block clock input.

The only difference in operation between Sampled, Non-Filtered (# 3A) and Sampled, Filtered (# 4A) is that, now, CR0[FILTER_CNT]>1, which activates filter operation.

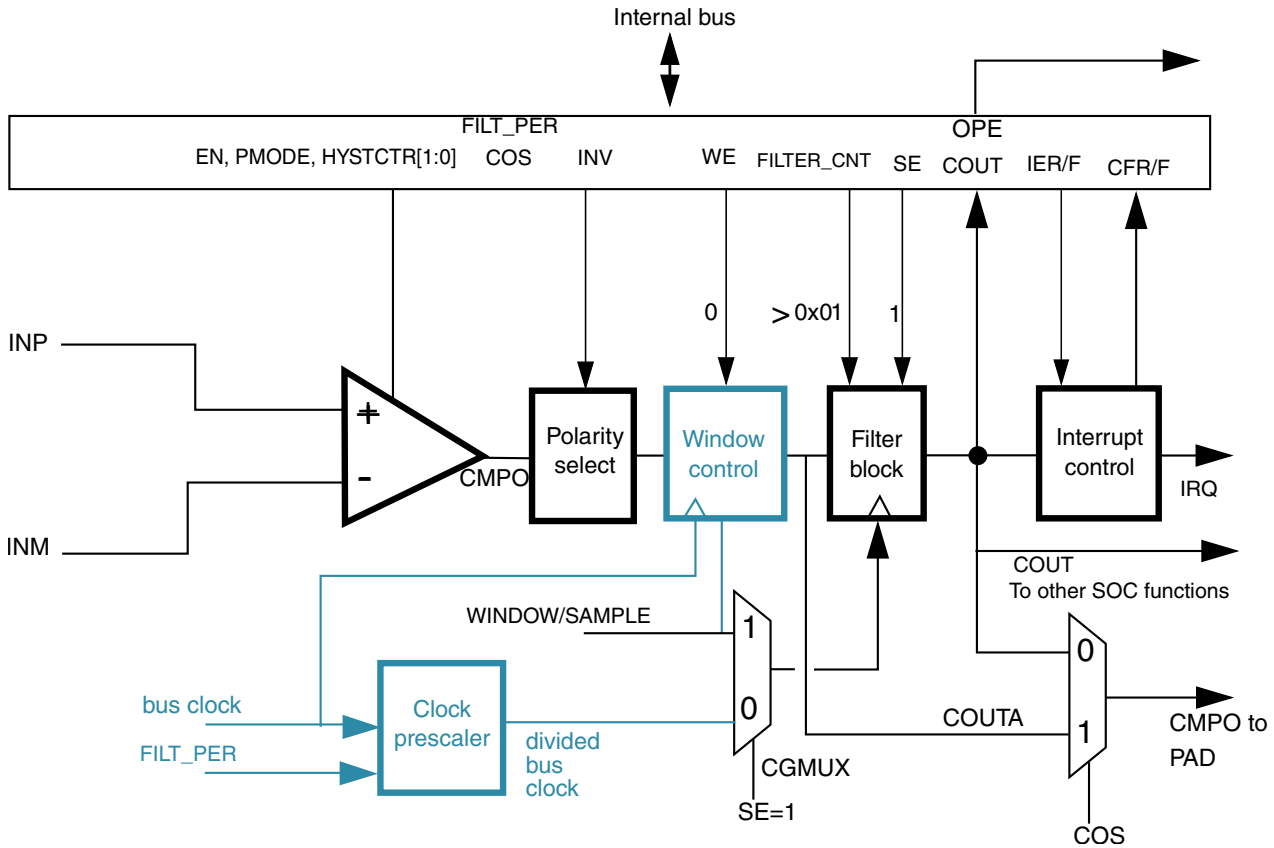


Figure 26-6. Sampled, Filtered (# 4A): sampling point externally driven

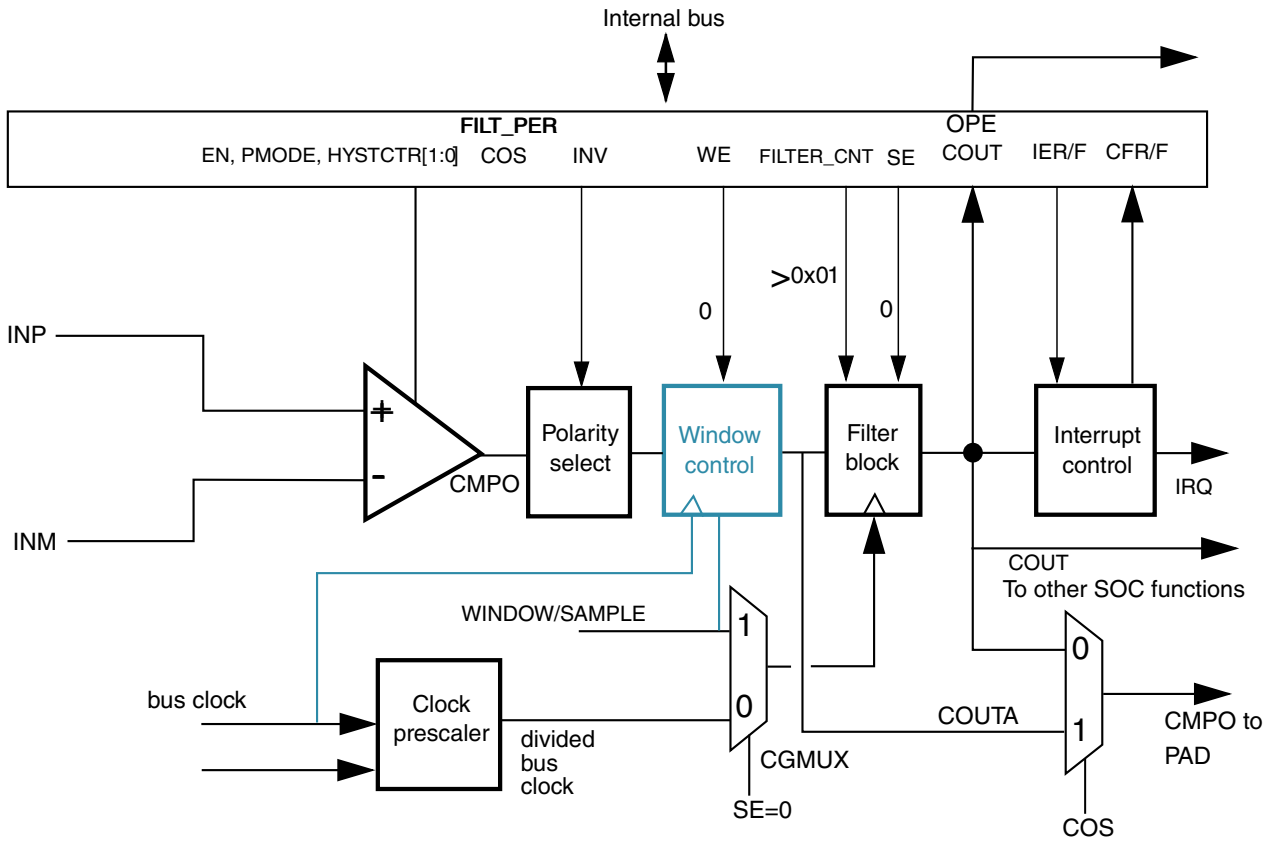


Figure 26-7. Sampled, Filtered (# 4B): sampling point internally derived

The only difference in operation between Sampled, Non-Filtered (# 3B) and Sampled, Filtered (# 4B) is that now, CR0[FILTER_CNT]>1, which activates filter operation.

26.3.1.5 Windowed mode (#s 5A & 5B)

The following figure illustrates comparator operation in the Windowed mode, ignoring latency of the analog comparator, polarity select, and window control block. It also assumes that the polarity select is set to non-inverting state.

NOTE

The analog comparator output is passed to COUTA only when the WINDOW signal is high.

In actual operation, COUTA may lag the analog inputs by up to one bus clock cycle plus the combinational path delay through the comparator and polarity select logic.

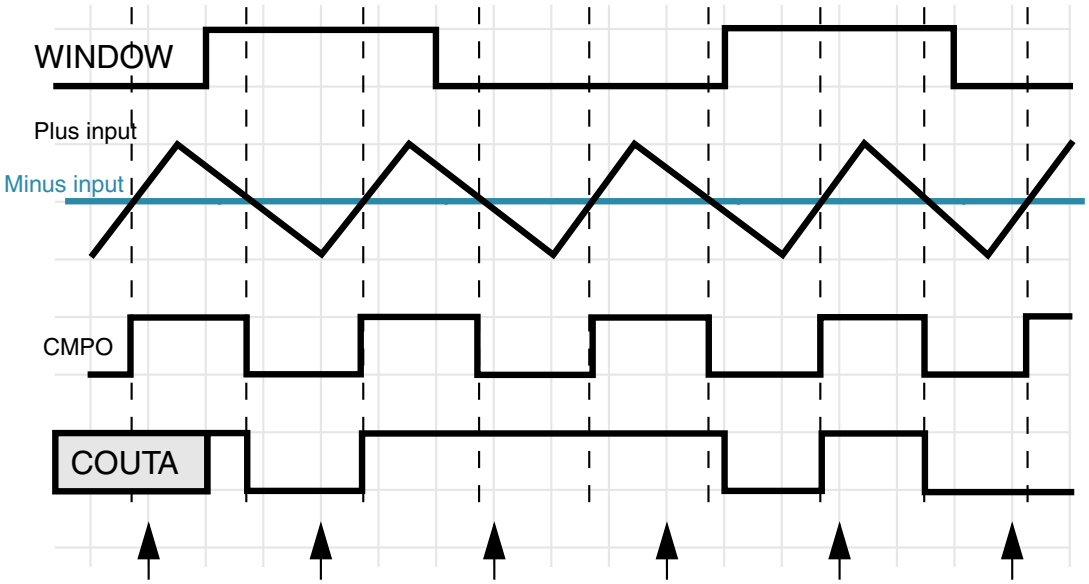


Figure 26-8. Windowed mode operation

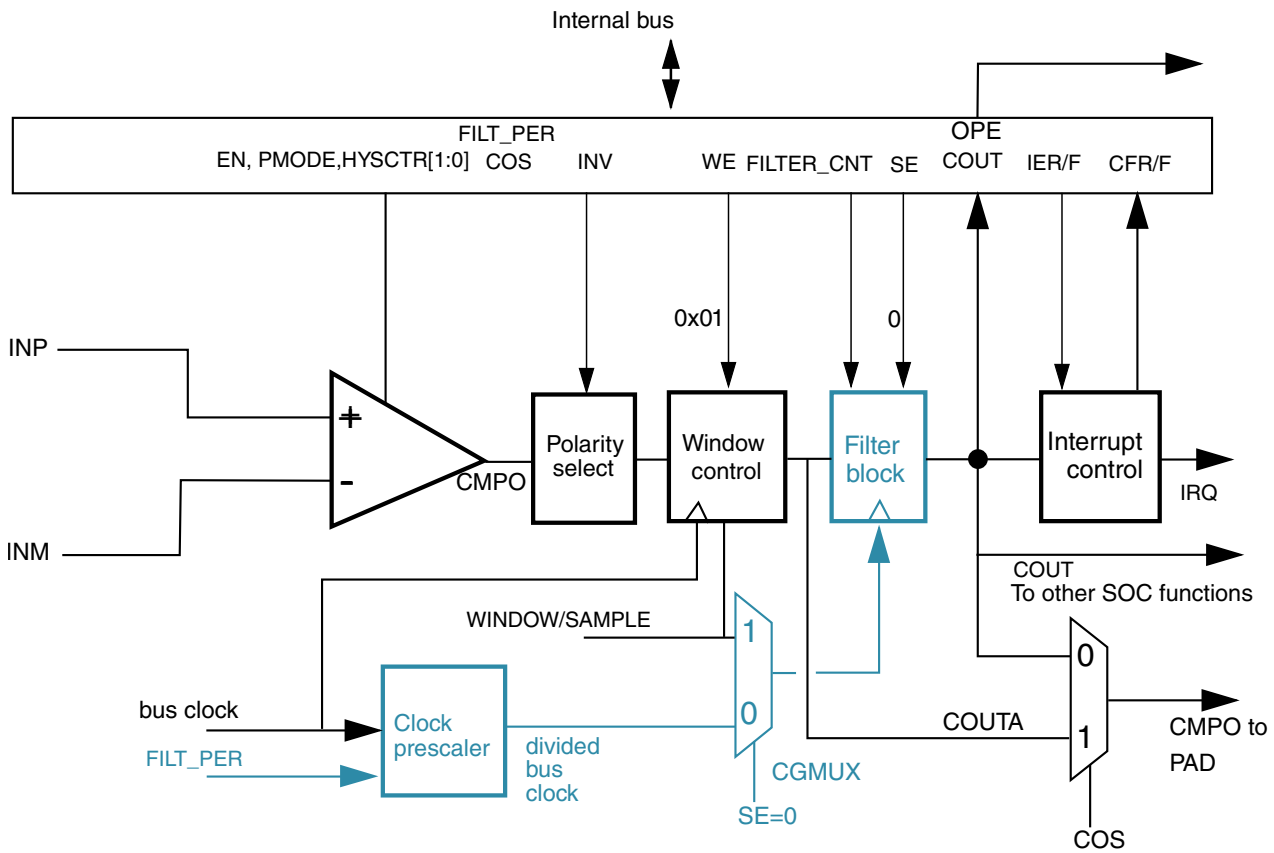


Figure 26-9. Windowed mode

For control configurations which result in disabling the filter block, see Filter Block Bypass Logic diagram.

When any windowed mode is active, COUTA is clocked by the bus clock whenever WINDOW = 1. The last latched value is held when WINDOW = 0.

26.3.1.6 Windowed/Resampled mode (# 6)

The following figure uses the same input stimulus shown in Figure 26-8, and adds resampling of COUTA to generate COUT. Samples are taken at the time points indicated by the arrows in the figure. Again, prop delays and latency are ignored for the sake of clarity.

This example was generated solely to demonstrate operation of the comparator in windowed/resampled mode, and does not reflect any specific application. Depending upon the sampling rate and window placement, COUT may not see zero-crossing events detected by the analog comparator. Sampling period and/or window placement must be carefully considered for a given application.

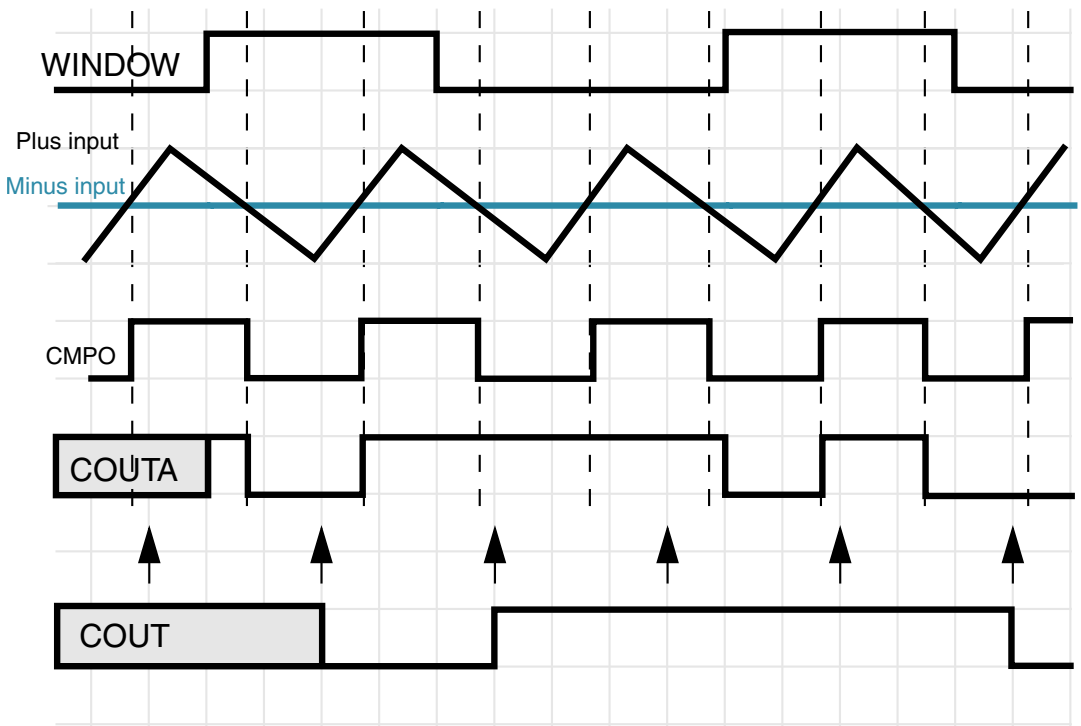


Figure 26-10. Windowed/resampled mode operation

This mode of operation results in an unfiltered string of comparator samples where the interval between the samples is determined by FPR[FILT_PER] and the bus clock rate. Configuration for this mode is virtually identical to that for the Windowed/Filtered Mode shown in the next section. The only difference is that the value of CR0[FILTER_CNT] must be 1.

26.3.1.7 Windowed/Filtered mode (#7)

This is the most complex mode of operation for the comparator block, as it uses both windowing and filtering features. It also has the highest latency of any of the modes. This can be approximated: up to 1 bus clock synchronization in the window function + ((CR0[FILTER_CNT] * FPR[FILT_PER]) + 1) * bus clock for the filter function.

When any windowed mode is active, COUTA is clocked by the bus clock whenever WINDOW = 1. The last latched value is held when WINDOW = 0.

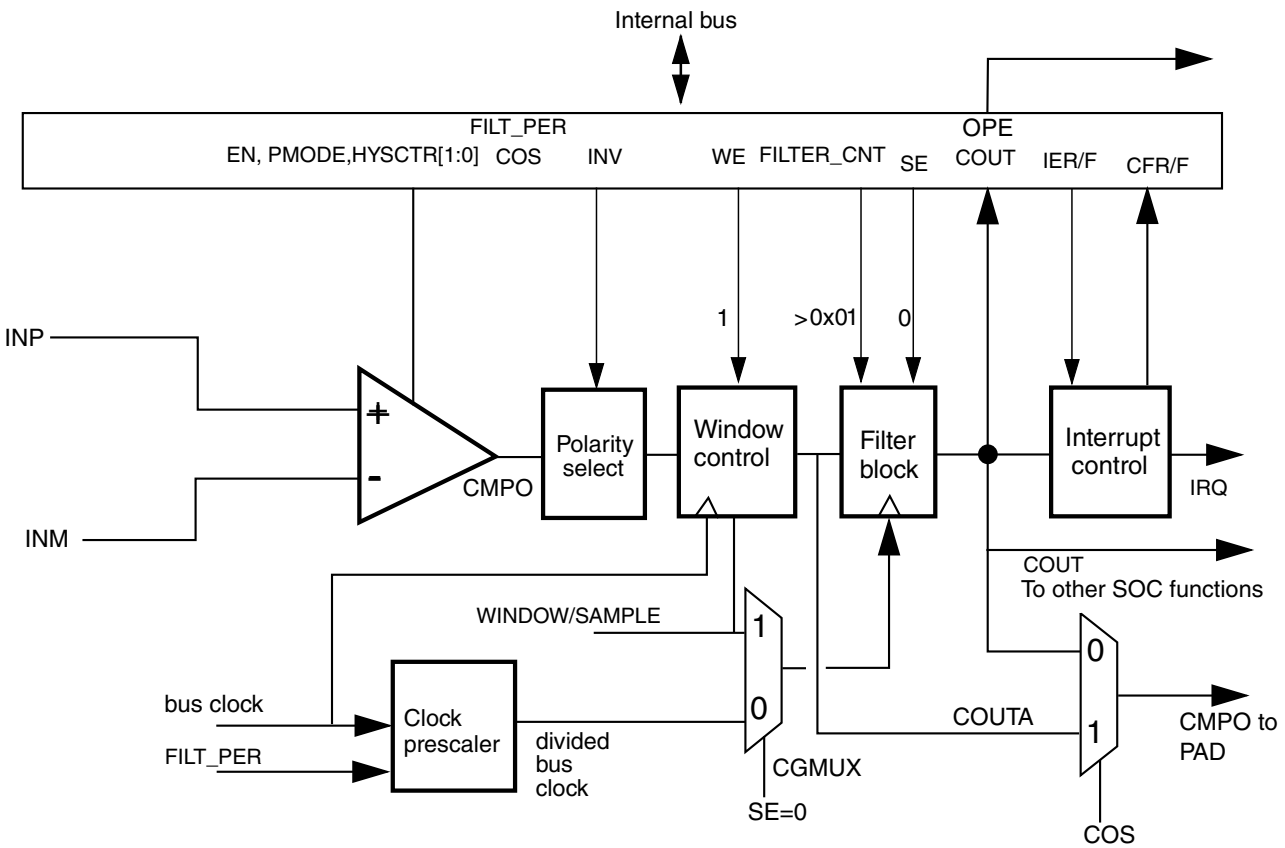


Figure 26-11. Windowed/Filtered mode

26.3.2 Power modes

26.3.2.1 Wait mode operation

During Wait and VLPW modes, the CMP, if enabled, continues to operate normally and a CMP interrupt can wake the MCU.

26.3.2.2 Stop mode operation

Depending on clock restrictions related to the MCU core or core peripherals, the MCU is brought out of stop when a compare event occurs and the corresponding interrupt is enabled. Similarly, if CR1[OPE] is enabled, the comparator output operates as in the normal operating mode and comparator output is placed onto the external pin. In Stop modes, the comparator can be operational in both:

- High-Speed (HS) Comparison mode when CR1[PMODE] = 1
- Low-Speed (LS) Comparison mode when CR1[PMODE] = 0

It is recommended to use the LS mode to minimize power consumption.

If stop is exited with a reset, all comparator registers are put into their reset state.

26.3.3 Startup and operation

A typical startup sequence is listed here.

- The time required to stabilize COUT will be the power-on delay of the comparators plus the largest propagation delay from a selected analog source through the analog comparator, windowing function and filter. See the Data Sheets for power-on delays of the comparators. The windowing function has a maximum of one bus clock period delay. The filter delay is specified in the Low-pass filter.
- During operation, the propagation delay of the selected data paths must always be considered. It may take many bus clock cycles for COUT and SCR[CFR]/SCR[CFF] to reflect an input change or a configuration change to one of the components involved in the data path.
- When programmed for filtering modes, COUT will initially be equal to 0, until sufficient clock cycles have elapsed to fill all stages of the filter. This occurs even if COUTA is at a logic 1.

26.3.4 Low-pass filter

The low-pass filter operates on the unfiltered and unsynchronized and optionally inverted comparator output COUTA and generates the filtered and synchronized output COUT.

Both COUTA and COUT can be configured as module outputs and are used for different purposes within the system.

Synchronization and edge detection are always used to determine status register bit values. They also apply to COUT for all sampling and windowed modes. Filtering can be performed using an internal timebase defined by FPR[FILT_PER], or using an external SAMPLE input to determine sample time.

The need for digital filtering and the amount of filtering is dependent on user requirements. Filtering can become more useful in the absence of an external hysteresis circuit. Without external hysteresis, high-frequency oscillations can be generated at COUTA when the selected INM and INP input voltages differ by less than the offset voltage of the differential comparator.

26.3.4.1 Enabling filter modes

Filter modes can be enabled by:

- Setting CR0[FILTER_CNT] > 0x01 and
- Setting FPR[FILT_PER] to a nonzero value or setting CR1[SE]=1

If using the divided bus clock to drive the filter, it will take samples of COUTA every FPR[FILT_PER] bus clock cycles.

The filter output will be at logic 0 when first initalized, and will subsequently change when all the consecutive CR0[FILTER_CNT] samples agree that the output value has changed. In other words, SCR[COUT] will be 0 for some initial period, even when COUTA is at logic 1.

Setting both CR1[SE] and FPR[FILT_PER] to 0 disables the filter and eliminates switching current associated with the filtering process.

Note

Always switch to this setting prior to making any changes in filter parameters. This resets the filter to a known state. Switching CR0[FILTER_CNT] on the fly without this intermediate step can result in unexpected behavior.

If CR1[SE]=1, the filter takes samples of COUTA on each positive transition of the sample input. The output state of the filter changes when all the consecutive CR0[FILTER_CNT] samples agree that the output value has changed.

26.3.4.2 Latency issues

The value of FPR[FILT_PER] or SAMPLE period must be set such that the sampling period is just longer than the period of the expected noise. This way a noise spike will corrupt only one sample. The value of CR0[FILTER_CNT] must be chosen to reduce the probability of noisy samples causing an incorrect transition to be recognized. The probability of an incorrect transition is defined as the probability of an incorrect sample raised to the power of CR0[FILTER_CNT].

The values of FPR[FILT_PER] or SAMPLE period and CR0[FILTER_CNT] must also be traded off against the desire for minimal latency in recognizing actual comparator output transitions. The probability of detecting an actual output change within the nominal latency is the probability of a correct sample raised to the power of CR0[FILTER_CNT].

The following table summarizes maximum latency values for the various modes of operation *in the absence of noise*. Filtering latency is restarted each time an actual output transition is masked by noise.

Table 26-2. Comparator sample/filter maximum latencies

Mode #	CR1[EN]	CR1[WE]	CR1[SE]	CR0[FILTER_CNT]	FPR[FILT_PER]	Operation	Maximum latency[1]
1	0	X	X	X	X	Disabled	N/A
2A	1	0	0	0x00	X	Continuous Mode	T_{PD}
2B	1	0	0	X	0x00		
3A	1	0	1	0x01	X	Sampled, Non-Filtered mode	$T_{PD} + T_{SAMPLE} + T_{per}$
3B	1	0	0	0x01	> 0x00		T_{PD} + (FPR[FILT_PER] * T_{per}) + T_{per}
4A	1	0	1	> 0x01	X	Sampled, Filtered mode	T_{PD} + (CR0[FILTER_CNT] * T_{SAMPLE}) + T_{per}
4B	1	0	0	> 0x01	> 0x00		T_{PD} + (CR0[FILTER_CNT] * FPR[FILT_PER] x T_{per}) + T_{per}
5A	1	1	0	0x00	X	Windowed mode	$T_{PD} + T_{per}$
5B	1	1	0	X	0x00		$T_{PD} + T_{per}$
6	1	1	0	0x01	0x01 - 0xFF	Windowed / Resampled mode	T_{PD} + (FPR[FILT_PER] * T_{per}) + $2T_{per}$
7	1	1	0	> 0x01	0x01 - 0xFF	Windowed / Filtered mode	T_{PD} + (CR0[FILTER_CNT] * FPR[FILT_PER] x T_{per}) + $2T_{per}$

1. T_{PD} represents the intrinsic delay of the analog component plus the polarity select logic. T_{SAMPLE} is the clock period of the external sample clock. T_{per} is the period of the bus clock.

26.4 CMP interrupts

The CMP module is capable of generating an interrupt on either the rising- or falling-edge of the comparator output, or both.

The following table gives the conditions in which the interrupt request is asserted and deasserted.

When	Then
SCR[IER] and SCR[CFR] are set	The interrupt request is asserted
SCR[IEF] and SCR[CFF] are set	The interrupt request is asserted
SCR[IER] and SCR[CFR] are cleared for a rising-edge interrupt	The interrupt request is deasserted
SCR[IEF] and SCR[CFF] are cleared for a falling-edge interrupt	The interrupt request is deasserted

26.5 Digital-to-analog converter

The figure found here shows the block diagram of the DAC module.

It contains a 64-tap resistor ladder network and a 64-to-1 multiplexer, which selects an output voltage from one of 64 distinct levels that outputs from DACO. It is controlled through the DAC Control Register (DACCR). Its supply reference source can be selected from two sources V_{in1} and V_{in2}. The module can be powered down or disabled when not in use. When in Disabled mode, DACO is connected to the analog ground.

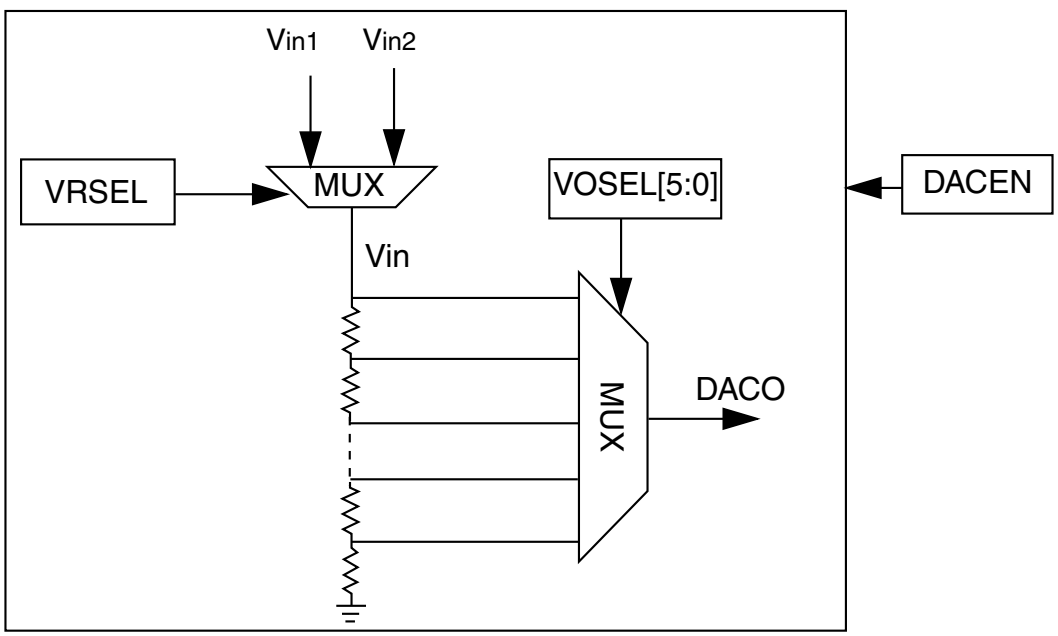


Figure 26-12. 6-bit DAC block diagram

26.6 DAC functional description

This section provides DAC functional description information.

26.6.1 Voltage reference source select

- V_{in1} connects to the primary voltage source as supply reference of 64 tap resistor ladder
- V_{in2} connects to an alternate voltage source

26.7 DAC resets

This module has a single reset input, corresponding to the chip-wide peripheral reset.

26.8 DAC clocks

This module has a single clock input, the bus clock.

26.9 DAC interrupts

This module has no interrupts.

Chapter 27
12-bit Digital-to-Analog Converter (DAC)

27.1 Introduction

27.1.1 Overview

The 12-bit digital-to-analog converter (DAC) provides a voltage reference to on-chip modules or an output to a package pin. It can also be used as a waveform generator to generate square, triangle, and sawtooth waveforms. The DAC can be put in powerdown mode if needed.

27.1.2 Features

DAC features include:

- 12-bit resolution
- Powerdown mode
- Output can be routed to the internal comparator or off-chip
- Choice of asynchronous or synchronous updates

  (sync input can be connected to internal crossbar)
- Automatic mode to generate square, triangle, and sawtooth output waveforms
- Automatic mode to allow programmable period, update rate, and range
- DMA support with configurable watermark level
- High-speed/low-speed mode selection

- Support of two digital formats
- Glitch filter to suppress output glitch during data conversion

27.1.3 Block Diagram

The following figure illustrates the DAC configuration.

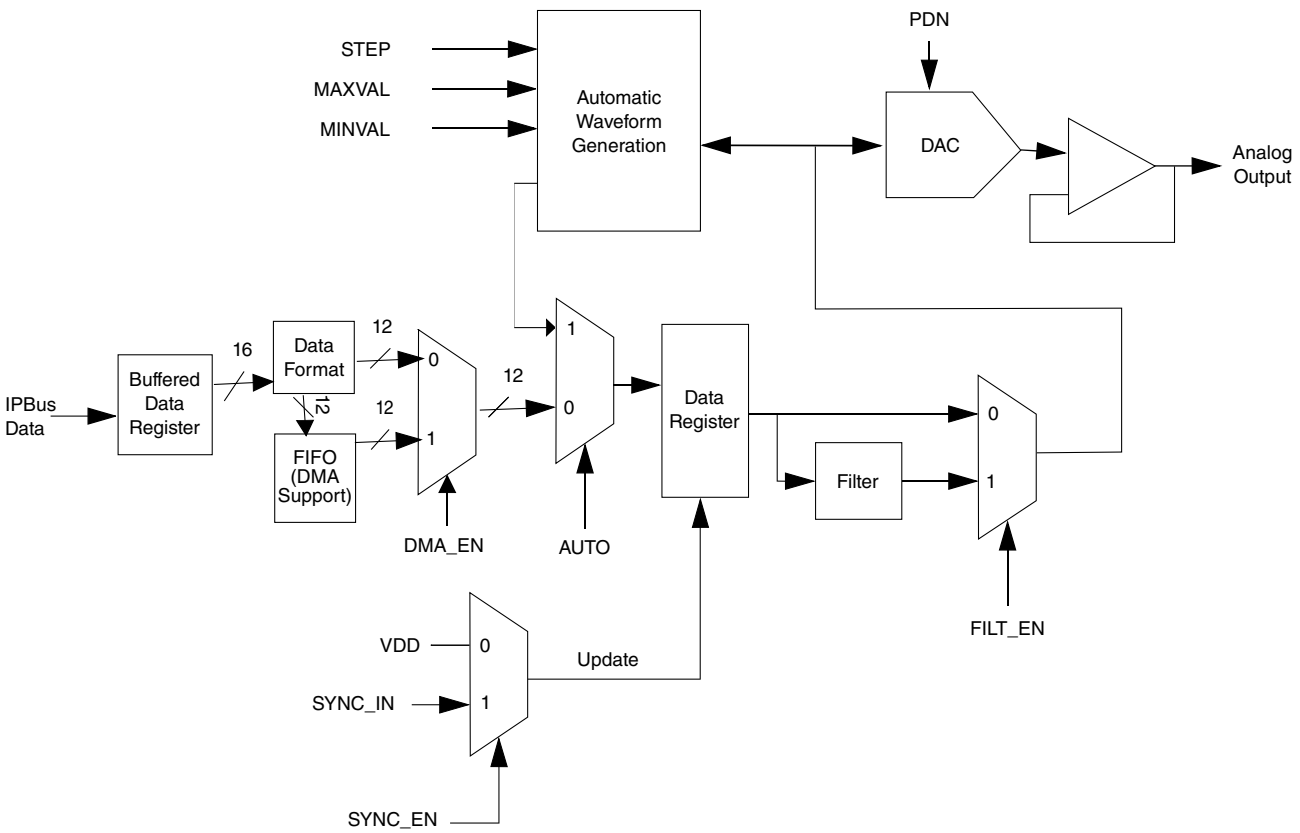


Figure 27-1. DAC Block Diagram

27.2 Memory Map and Registers

The address of a register is the sum of a base address and an address offset. The base address is defined at the chip level. The address offset is defined at the module level. Only 16-bit accesses are supported; no 8-bit or 32-bit accesses can be done.

DAC memory map

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
E000	Control Register 0 (DAC_CTRL0)	16	R/W	1101h	27.2.1/607
E001	Buffered Data Register (DAC_DATAREG_FMT0)	16	R/W	0000h	27.2.2/609
E001	Buffered Data Register (DAC_DATAREG_FMT1)	16	R/W	0000h	27.2.3/610
E002	Step Size Register (DAC_STEPVAL_FMT0)	16	R/W	0000h	27.2.4/610
E002	Step Size Register (DAC_STEPVAL_FMT1)	16	R/W	0000h	27.2.5/611
E003	Minimum Value Register (DAC_MINVAL_FMT0)	16	R/W	0000h	27.2.6/611
E003	Minimum Value Register (DAC_MINVAL_FMT1)	16	R/W	0000h	27.2.7/612
E004	Maximum Value Register (DAC_MAXVAL_FMT0)	16	R/W	FFFFh	27.2.8/612
E004	Maximum Value Register (DAC_MAXVAL_FMT1)	16	R/W	FFFFh	27.2.9/613
E005	Status Register (DAC_STATUS)	16	R	0001h	27.2.10/ 613
E006	Control Register 1 (DAC_CTRL1)	16	R/W	001Dh	27.2.11/ 614

27.2.1 Control Register 0 (DAC_CTRL0)

Address: E000h base + 0h offset = E000h

Bit	15	14	13	12	11	10	9	8
Read	0			FILT_EN	0		WTMK_LVL	
Write								
Reset	0	0	0	1	0	0	0	1

Bit	7	6	5	4	3	2	1	0
Read Write	DMA_EN	HSLS	UP	DOWN	AUTO	SYNC_EN	FORMAT	PDN
Reset	0	0	0	0	0	0	0	1

DAC_CTRL0 field descriptions

Field	Description
15–13 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
12 FILT_EN	Glitch Filter Enable This bit enables the glitch suppression filter. This introduces a latency equivalent to FILT_CNT IPBus clock cycles for DAC output updates. It should never be low. 0 Filter disabled. 1 Filter enabled.
11–10 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
9–8 WTMK_LVL	Watermark Level

Table continues on the next page...

DAC_CTRL0 field descriptions (continued)

Field	Description
	When the level of FIFO is less than or equal to this field, a DMA request should be sent. The FIFO used for DMA support generates a watermark signal depending on the value of WTMK_LVL, which is used for asserting a DMA request . 00 Watermark value is 0 01 Watermark value is 2 (default) 10 Watermark value is 4 11 Watermark value is 6
7 DMA_EN	Enable DMA Support This bit enables DMA support. When DMA support is enabled, the data to be presented to the analog DAC input is fetched from the FIFO. If SYNC_EN is set, then the data is read from the FIFO on the SYNC_IN trigger, and the data read from the FIFO is presented to the analog DAC input on the delayed trigger of SYNC_IN. If SYNC_EN is not set, then data is read from the FIFO and presented to the analog DAC every clock cycle. The data is written to the FIFO (used for DMA support) on every clock. **Restriction:** DMA_EN should never be set when SYNC_EN is low and cannot be used unless SYS_CLK is very slow. Therefore SYNC_EN should always be set when DMA_EN is set. 0 DMA support disabled (default) 1 DMA support enabled
6 HSLS	High/Low Speed This bit gives flexibility to the user to choose between speed and power while operating in normal mode. Setting this input low selects high speed mode, in which the settling time of the DAC is 1 µs (faster response) but at the expense of higher power consumption. Setting the bit high selects low speed mode, which saves power but in which the DAC takes more time to settle down. 0 High speed mode (default) 1 Low speed mode
5 UP	Enable Up Counting This bit enables counting up in automatic mode. See the functional description of automatic mode to understand how this bit affects automatic waveform generation. 0 Up counting disabled. 1 Up counting enabled.
4 DOWN	Enable Down Counting This bit enables counting down in automatic mode. See the functional description of automatic mode to understand how this bit affects automatic waveform generation. 0 Down counting disabled. 1 Down counting enabled.
3 AUTO	Automatic Mode This bit enables automatic waveform generation mode. In automatic mode an external source (typically a timer module) driving SYNC_IN determines the data update rate while the STEP, MINVAL, and MAXVAL registers and the UP and DOWN bits are used to shape the waveform. If the SYNC_EN bit is not set when using this mode, then the data for the analog DAC will be updated every clock cycle, but the DAC output may be unable to keep up with this update rate. **Restriction:** AUTO should never be set when SYNC_EN is low and cannot be used unless ipg_clk is very slow.

Table continues on the next page...

DAC_CTRL0 field descriptions (continued)

Field	Description
	0 Normal mode. Automatic waveform generation disabled. 1 Automatic waveform generation enabled.
2 SYNC_EN	Sync Enable This bit enables the SYNC_IN input to be used to trigger an update of the buffered data being presented to the analog DAC input. If SYNC_EN is clear, then asynchronous mode is indicated and data written to the buffered data register is presented to the analog DAC input in the following IPBus clock cycle. **Restriction:** Do not set SYNC_EN low when DMA_EN is set high. 0 Asynchronous mode. Data written to the buffered data register is presented to DAC inputs on the next clock cycle. 1 Synchronous mode. Rising edge of SYNC_IN updates data in the buffered data register presented to the DAC input.
1 FORMAT	Data Format Two data formats can be used for the DAC. When this bit is clear, the 12 bits of data are right justified within the 16-bit data register. When this bit is set, the 12 bits of data are left justified. In either case, the 4 unused bits are ignored. 0 Data words are right justified. (default) 1 Data words are left justified.
0 PDN	Power Down This bit is used to power down the analog portion of the DAC (resulting in its output being pulled low) when it is not in use. This bit does not reset the registers; upon clearing PDN, the analog DAC will output the value currently presented to its inputs. The analog block requires 12 µs to recover from the power down state before proper operation is guaranteed. 0 DAC is operational. 1 DAC is powered down. (default)

27.2.2 Buffered Data Register (DAC_DATAREG_FMT0)

Address: E000h base + 1h offset = E001h

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	0				DATA											
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

DAC_DATAREG_FMT0 field descriptions

Field	Description
15–12 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
DATA	DAC data (right justified) Data written to this register is held in a buffer. The digital data contained in this buffer is presented to the analog DAC upon the rising edge of the SYNC_IN signal (or at the next clock cycle if CTRL0[SYNC_EN] is

Table continues on the next page...

DAC_DATAREG_FMT0 field descriptions (continued)

Field	Description
	clear) and converted to analog and output by the DAC. Reading this register returns the value being presented to the analog DAC, which may differ from the data value being held in the write buffer. The data in this buffer can be updated at any rate, but the DAC output load impedance may affect the updating rate . When a read occurs at the address of this register, the data read is the value of the data presented to the analog DAC at that time, not the value in this register.

27.2.3 Buffered Data Register (DAC_DATAREG_FMT1)

Address: E000h base + 1h offset = E001h

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	DATA												0			
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

DAC_DATAREG_FMT1 field descriptions

Field	Description
15–4 DATA	DAC data (left justified) Data written to this register is held in a buffer. The digital data contained in this buffer is presented to the analog DAC upon the rising edge of the SYNC_IN signal (or at the next clock cycle if CTRL0[SYNC_EN] is clear) and converted to analog and output by the DAC. Reading this register returns the value being presented to the analog DAC, which may differ from the data value being held in the write buffer. The data in this buffer can be updated at any rate, but the DAC output load impedance may affect the updating rate . When a read occurs at the address of this register, the data read is the value of the data presented to the analog DAC at that time, not the value in this register.
Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

27.2.4 Step Size Register (DAC_STEPVAL_FMT0)

Address: E000h base + 2h offset = E002h

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	0				STEP											
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

DAC_STEPVAL_FMT0 field descriptions

Field	Description
15–12 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
STEP	STEP size (right justified)

Table continues on the next page...

DAC_STEPVAL_FMT0 field descriptions (continued)

Field	Description
	When the DAC is in automatic mode (CTRL0[AUTO] = 1), the step size contained in this register will be added to or subtracted from the current value to create the next value presented to the DAC inputs. This register is not used during normal mode operation, but can still be written to and read from.

27.2.5 Step Size Register (DAC_STEPVAL_FMT1)

Address: E000h base + 2h offset = E002h

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	STEP												0			
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

DAC_STEPVAL_FMT1 field descriptions

Field	Description
15–4 STEP	STEP size (left justified) When the DAC is in automatic mode (CTRL0[AUTO] = 1), the step size contained in this register will be added to or subtracted from the current value to create the next value presented to the DAC inputs. This register is not used during normal mode operation, but can still be written to and read from.
Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

27.2.6 Minimum Value Register (DAC_MINVAL_FMT0)

Address: E000h base + 3h offset = E003h

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	0				MINVAL											
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

DAC_MINVAL_FMT0 field descriptions

Field	Description
15–12 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
MINVAL	Minimum value (right justified) When the DAC is in automatic mode (CTRL0[AUTO] = 1), the minimum value contained in this register acts as the lower range limit during automatic waveform generation. This register is not used during normal mode operation, but can still be written to and read from. Refer to the device data sheet for limitations on the low end voltage output of the DAC. **NOTE:** If DAC input data is less than MINVAL, output is limited to MINVAL during automatic waveform generation.

27.2.7 Minimum Value Register (DAC_MINVAL_FMT1)

Address: E000h base + 3h offset = E003h

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	MINVAL												0			
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

DAC_MINVAL_FMT1 field descriptions

Field	Description
15–4 MINVAL	Minimum value (left justified) When the DAC is in automatic mode (CTRL0[AUTO] = 1), the minimum value contained in this register acts as the lower range limit during automatic waveform generation. This register is not used during normal mode operation, but can still be written to and read from. Refer to the device data sheet for limitations on the low end voltage output of the DAC. **NOTE:** If DAC input data is less than MINVAL, output is limited to MINVAL during automatic waveform generation.
Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

27.2.8 Maximum Value Register (DAC_MAXVAL_FMT0)

Address: E000h base + 4h offset = E004h

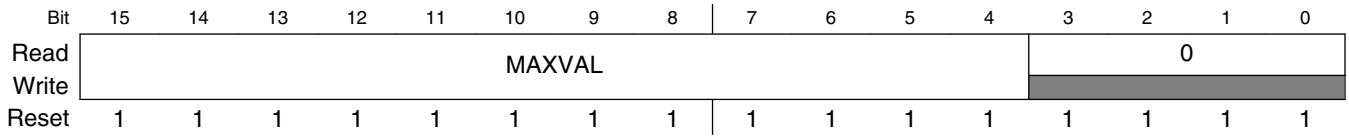
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	0				MAXVAL											
Write																
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

DAC_MAXVAL_FMT0 field descriptions

Field	Description
15–12 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
MAXVAL	Maximum value (right justified) When the DAC is in automatic mode (CTRL0[AUTO] = 1), the maximum value contained in this register acts as the upper range limit during automatic waveform generation. This register is not used during normal mode operation, but can still be written to and read from. Refer to the device data sheet for limitations on the high end voltage output of the DAC. **NOTE:** If DAC input data is greater than MAXVAL, output is limited to MAXVAL during automatic waveform generation.

27.2.9 Maximum Value Register (DAC_MAXVAL_FMT1)

Address: E000h base + 4h offset = E004h

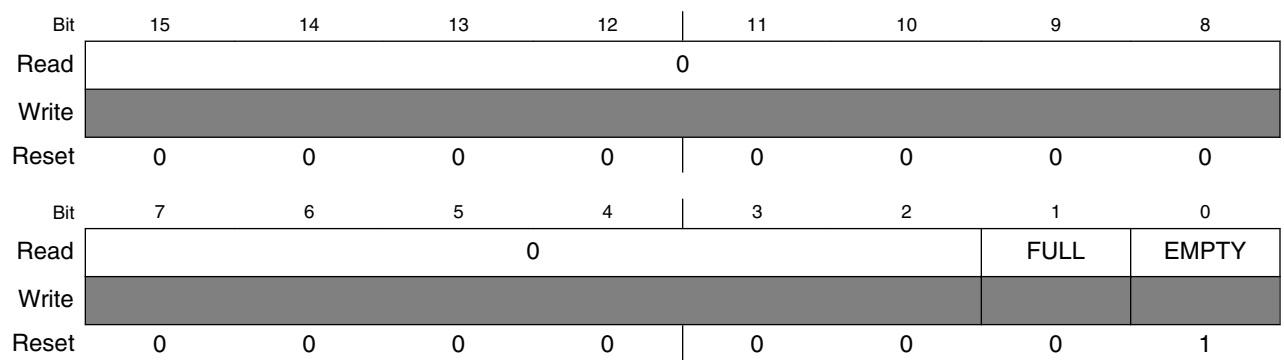


DAC_MAXVAL_FMT1 field descriptions

Field	Description
15–4 MAXVAL	Maximum value (left justified) When the DAC is in automatic mode (CTRL0[AUTO] = 1), the maximum value contained in this register acts as the upper range limit during automatic waveform generation. This register is not used during normal mode operation, but can still be written to and read from. Refer to the device data sheet for limitations on the high end voltage output of the DAC. **NOTE:** If DAC input data is greater than MAXVAL, output is limited to MAXVAL during automatic waveform generation.
Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

27.2.10 Status Register (DAC_STATUS)

Address: E000h base + 5h offset = E005h



DAC_STATUS field descriptions

Field	Description
15–2 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1 FULL	Indicates full status of FIFO 0 FIFO is not full (on reset). 1 FIFO is full.

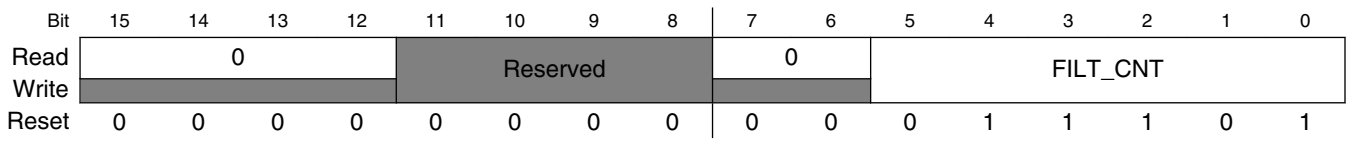
Table continues on the next page...

DAC_STATUS field descriptions (continued)

Field	Description
0 EMPTY	Indicates empty status of FIFO 0 FIFO is not empty. 1 FIFO is empty (on reset).

27.2.11 Control Register 1 (DAC_CTRL1)

Address: E000h base + 6h offset = E006h



DAC_CTRL1 field descriptions

Field	Description
15–12 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
11–8 Reserved	This field is reserved. Do not write to this bitfield.
7–6 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
FILT_CNT	Glitch Filter Count This field represents the number of IP Bus clock cycles for which the DAC output is held unchanged after new data is presented to the analog DAC's inputs. The number of clock cycles for which DAC output is held unchanged is equal to the value of FILT_CNT. Approximately 240 ns is needed for worst-case settling of the DAC output, so a value of 29 should be used for 125 MHz operation, a value of 33 for 140 MHz operation, and a value of 12 for 50 MHz operation. The default value is 29 (can be used for 125 MHz operation). **NOTE:** When using the glitch filter, ensure that the filter count is less than the update count. Otherwise, the DAC output will never be updated.

27.3 Functional Description

27.3.1 Conversion modes

This DAC supports two conversion modes: asynchronous conversion and synchronous conversion.

27.3.1.1 Asynchronous conversion mode

Data can be immediately presented to the DAC and converted to an analog output when it is written to the DAC buffered data register.

27.3.1.2 Synchronous conversion mode

Data in the DAC buffered data register is controlled by the SYNC_IN signal when the buffered data is presented to the input of the DAC. The update occurs on the rising edge of the SYNC_IN signal. The SYNC_IN signal can come from a timer, comparators, pins, or other sources. The CPU needs to update the buffered data register prior to the next SYNC_IN rising edge. Otherwise, the old buffered data is reused. Note: The SYNC_IN single must be high for at least one IPBus clock cycle and must be low for at least one IPBus clock cycle.

27.3.2 Operation Modes

The DAC operates in either Normal or Automatic mode. In Normal mode, it generates an analog representation of digital words. In Automatic mode, it generates sawtooth, triangle, and square waveforms without CPU intervention.

27.3.2.1 Normal Mode

The DAC receives data words through a memory-mapped register on the IPBus (DATA). A digital word is applied to the DAC inputs based on CTRL[SYNC_EN]. In the worst case with no DAC output load, approximately 240 ns settling time is needed.

27.3.2.2 DMA Support Mode

This mode uses a FIFO for data to be presented to an analog DAC input. The data received on the IPBus (DATA) is written to the FIFO on every clock edge. The data is read from the FIFO and applied to the DAC inputs based on the SYNC_EN bit. If SYNC_EN is set, then the data is read from the FIFO on the rising edge of SYNC_IN, and the data read from the FIFO is presented to the analog DAC input on the delayed rising edge of SYNC_IN.

The FIFO used for DMA support also generates a watermark signal depending on the value of WTMK_LVL, which is used for asserting a DMA request . This request is de-asserted when transfer is acknowledged by the DMA controller, which signifies that the required data transfer is complete.

27.3.2.3 Automatic Mode

In Automatic mode, the DAC generates sawtooth, triangle, and square wave waveforms without CPU intervention. The update rate, incremental step size, and minimum and maximum values are programmable.

The value in the DATA register is used as a starting-point for the following process:

1. The SYNC_IN input indicates that it is time to update the data presented to the DAC.
2. The STEP value is added to/subtracted from the current DATA value and DATA is updated.
3. If CTRL[UP] is set, then STEP is added to DATA each update until MAXVAL is reached.
4. The generator starts subtracting STEP from DATA if CTRL[DOWN] is set (down counting enabled) or reloading MINVAL if CTRL[DOWN] is clear (no down counting).
5. When DATA reaches MINVAL while counting down, the generator starts counting up if CTRL[UP] is set or reloads MAXVAL if CTRL[UP] is clear (up counting disabled).

The initial count direction depends on which bit (CTRL[UP] or CTRL[DOWN]) is set last.

The following figures show examples of automatically-generated waveforms. The waveforms shown are ideal. Actual waveforms are limited by the slew rate of the DAC output.

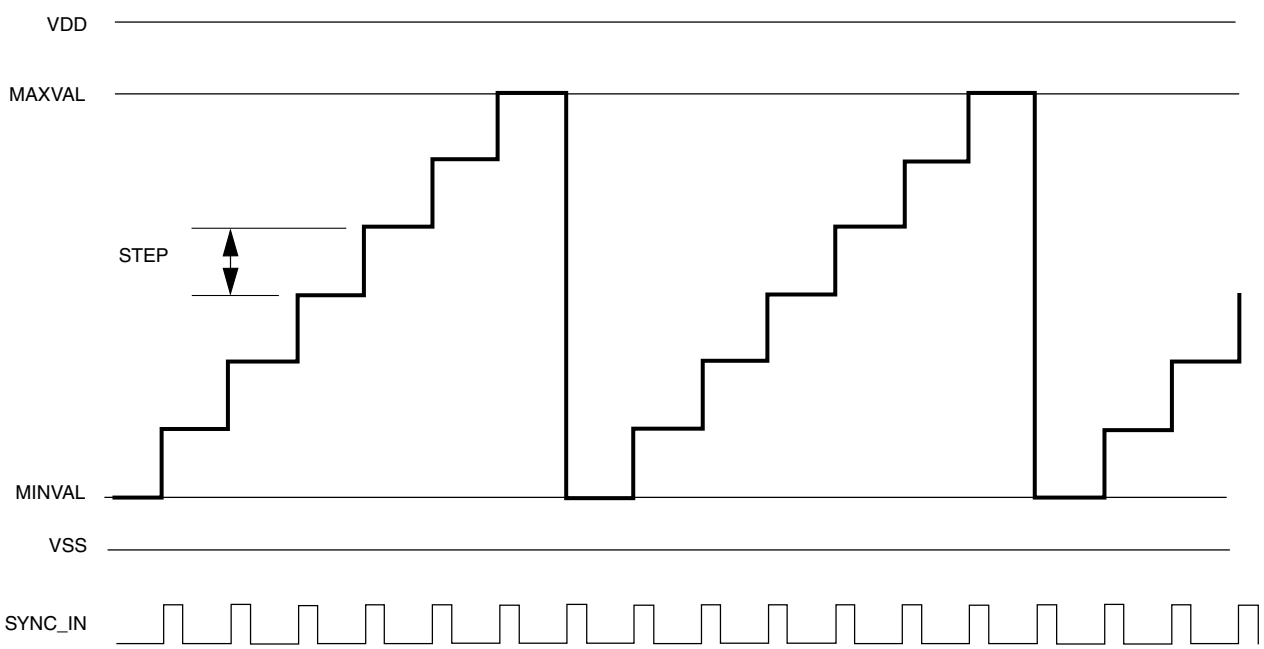


Figure 27-2. Sawtooth Waveform Example with CTRL[UP]=1 and CTRL[DOWN]=0

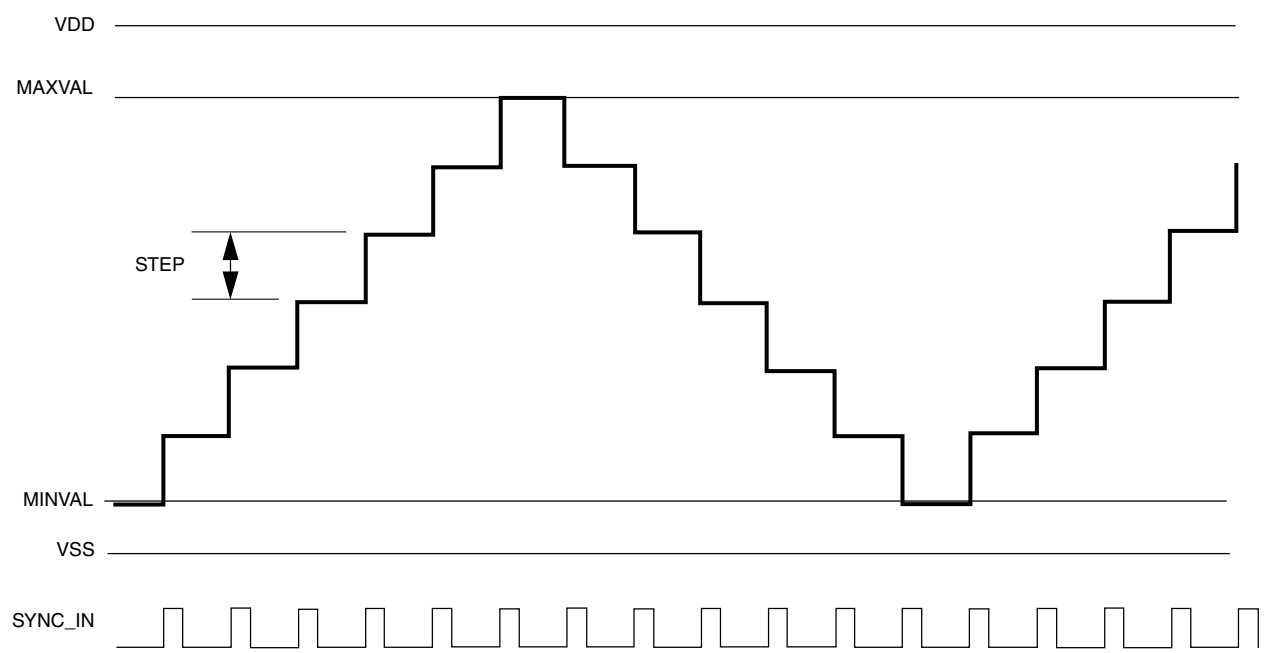


Figure 27-3. Triangle Waveform Example with CTRL[UP]=1 and CTRL[DOWN]=1

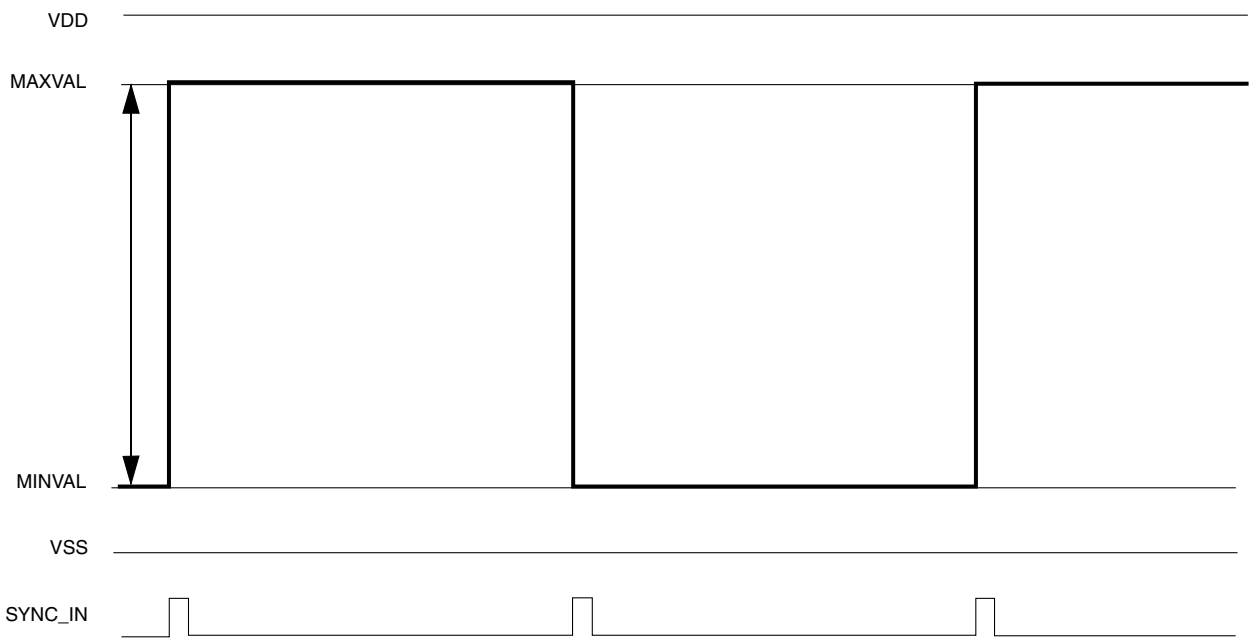


Figure 27-4. Square Waveform Example with CTRL[UP]=1 and CTRL[DOWN]=1

These examples show that the waveform period is a function of the difference between MAXVAL and MINVAL, the STEP size, and the update rate as shown here:

$$\text{Period} = \left[\frac{\text{MAXVAL} - \text{MINVAL}}{\text{STEP}} \times \text{UpdatePeriod}\right] \times 2$$

Increasing STEP decreases the resolution of the output steps. Increasing the update rate decreases the waveform period. Varying MINVAL and MAXVAL changes the DC offset and the amplitude of the waveform.

27.3.3 DAC settling time

Settling time is the interval, within a specified percentage error band, between the time data is presented to the DAC input to update (change) and the time its analog output value reaches its final value. Settling time is affected by circuit propagation delay and the slew rate of the DAC output capability, plus the real load on DAC output.

Approximately 240 ns settling time, which is caused by the DAC conversion glitch, is needed in the worst case situations when DAC output is internally fed to other modules, such as comparator inputs. If DAC output is to a package pin, the additional settling time is affected by the slew rate of an output amplifier and the load on the pin. The maximum settling time will not exceed 2 microseconds with a maximum output load (3 kohm || 400 pf) when the output swings from minimum output to maximum output or vice-versa.

27.3.4 Waveform Programming Example

To create a waveform that goes down from 3.0 V to 1.5 V in 1 millisecond, first calculate MAXVAL and MINVAL. Based on each DAC LSB representing 0.806 mV (assuming the DAC reference is 3.3 V), we find that MAXVAL is 0xE8A and MINVAL is 0x745. These numbers represent a difference of 1861 LSBs to be accomplished in 1 millisecond, so we can safely update the DAC 500 times because the DAC has a maximum 2-microsecond settling time. Programming calculations are as follows:

- To go from MAXVAL to MINVAL in 500 steps, STEP must be 0x004 after rounding the result of 1861/500.
- To go from MAXVAL to MINVAL with a STEP size of 0x004 requires 465.25 steps.
- To keep the waveform period of 1 millisecond using 465.25 updates requires an update period of 465.25 kHz.
- If the system clock operates at 100 MHz, program a timer module to pulse the SYNC_IN input every 100000000/465250 = 215 counts.

When the timer module is programmed along with the MAXVAL, MINVAL, and STEP registers, the DATA register is written with a value equal to MAXVAL as a starting point because the DAC is only down counting. When writing to the DATA, STEP, MAXVAL, and MINVAL registers, ensure that FORMAT has the desired value and that the data values are properly justified to match FORMAT. The SYNC_EN, DOWN, and AUTO bits of the CTRL register should be set. The CTRL[UP] and CTRL[PDN] bits are cleared. To suppress glitches on the output, set CTRL[FILT_CNT] and CTRL[FILT_EN]. The desired waveform starts within 12 microseconds from the clearing of CTRL[PDN] and continues until CTRL[PDN] is set or the timer is stopped.

27.3.5 Sources of Waveform Distortion

27.3.5.1 Switching Glitches

When a new digital value is presented to the DAC input, some glitches may appear on the output as the new values propagate through the circuitry. Eventually, these glitches settle out and the output slews to its new value. To avoid these glitches, set FILT_EN to 1 and give FILT_CNT a suitable value to cause the DAC to hold its current output for the number of clock cycles specified in the description of the FILT_CNT field, during which time the switching glitches settle out. After the filter time is satisfied, the output smoothly slews to the new value.

27.3.5.2 Slew Effects

The example waveforms are ideal waveforms and show transitions as step functions. In reality, the DAC output has a finite slew rate that rounds off the steps. Whether this rounding off is noticeable depends on the output step size (larger output changes require longer settling times) and on the update period (longer dwell times make the settling times less noticeable).

27.3.5.3 Clipping Effects (Automatic Mode Only)

One form of clipping occurs during automatic waveform generation when the difference between MAXVAL and MINVAL is not a (near) even multiple of the STEP value. This results in a partial step as the waveform approaches MAXVAL and MINVAL. The following figure shows an example.

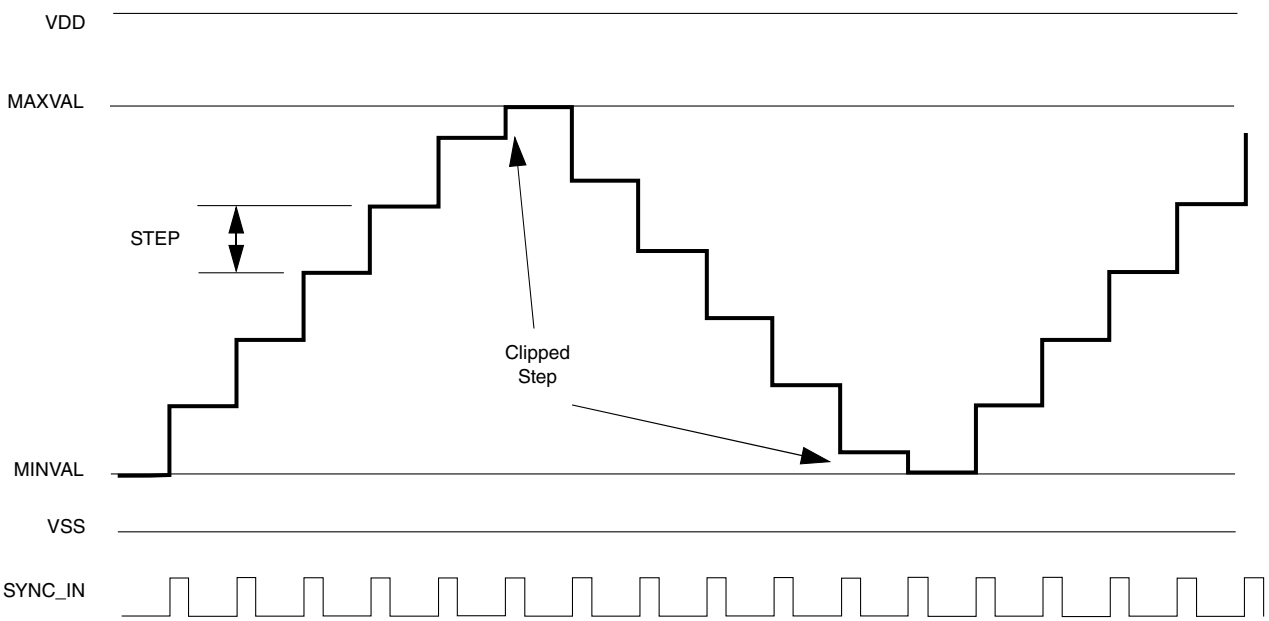


Figure 27-5. Triangle Waveform Example with Clipping

Another form of clipping occurs when the MAXVAL or MINVAL is beyond the output range of the DAC. The maximum and minimum voltages that can be driven out are defined in the device data sheet.

27.4 Resets

When reset, all of the registers return to the reset state.

27.5 Clocks

The DAC uses the system bus clock (IPBus clock).

27.6 Interrupts

The DAC module does not generate any interrupts.

Chapter 28
Enhanced Flexible Pulse Width Modulator A (PWMA)

28.1 Introduction

The pulse width modulator (PWM) module contains PWM submodules, each of which is set up to control a single half-bridge power stage. Fault channel support is provided.

This PWM module can generate various switching patterns, including highly sophisticated waveforms. It can be used to control all known Switched Mode Power Supplies (SMPS) topologies.

28.1.1 Features

- 16 bits of resolution for center, edge-aligned, and asymmetrical PWMs
- Fractional PWM clock generation for enhanced resolution of the PWM period and duty cycle
- PWM outputs that can operate as complementary pairs or independent channels
- Ability to accept signed numbers for PWM generation
- Independent control of both edges of each PWM output
- Support for synchronization to external hardware or other PWM
- Double buffered PWM registers
    - Integral reload rates from 1 to 16
    - Half cycle reload capability
- Multiple output trigger events can be generated per PWM cycle via hardware
- Support for double switching PWM outputs
- Fault inputs can be assigned to control multiple PWM outputs
- Programmable filters for fault inputs
- Independently programmable PWM output polarity
- Independent top and bottom deadtime insertion
- Each complementary pair can operate with its own PWM frequency and deadtime values
- Individual software control for each PWM output

- All outputs can be programmed to change simultaneously via a FORCE_OUT event
- PWM_X pin can optionally output a third PWM signal from each submodule
- Channels not used for PWM generation can be used for buffered output compare functions
- Channels not used for PWM generation can be used for input capture functions
- Enhanced dual edge capture functionality
- The option to supply the source for each complementary PWM signal pair from any of the following:
  - External digital pin
  - Internal timer channel
  - Crossbar module outputs
  - External ADC input, taking into account values set in ADC high and low limit registers

28.1.2 Modes of Operation

Be careful when using this module in stop, wait and debug operating modes.

CAUTION

Some applications require regular software updates for proper operation. Failure to provide regular software updates could result in destroying the hardware setup.

To accommodate this situation, PWM outputs are placed in their inactive states in stop mode, and they can optionally be placed in inactive states in wait and debug (EOnCE) modes. PWM outputs are reactivated (assuming they were active beforehand) when these modes are exited.

Table 28-1. Modes when PWM Operation is Restricted

Mode	Description
Stop	PWM outputs are inactive.
Wait	PWM outputs are driven or inactive as a function of CTRL2[WAITEN].
Debug	CPU and peripheral clocks continue to run, but the CPU may stall for periods of time. PWM outputs are driven or inactive as a function of CTRL2[DBGEN].

28.1.3 Block Diagram

The following figure is a block diagram of the PWM.

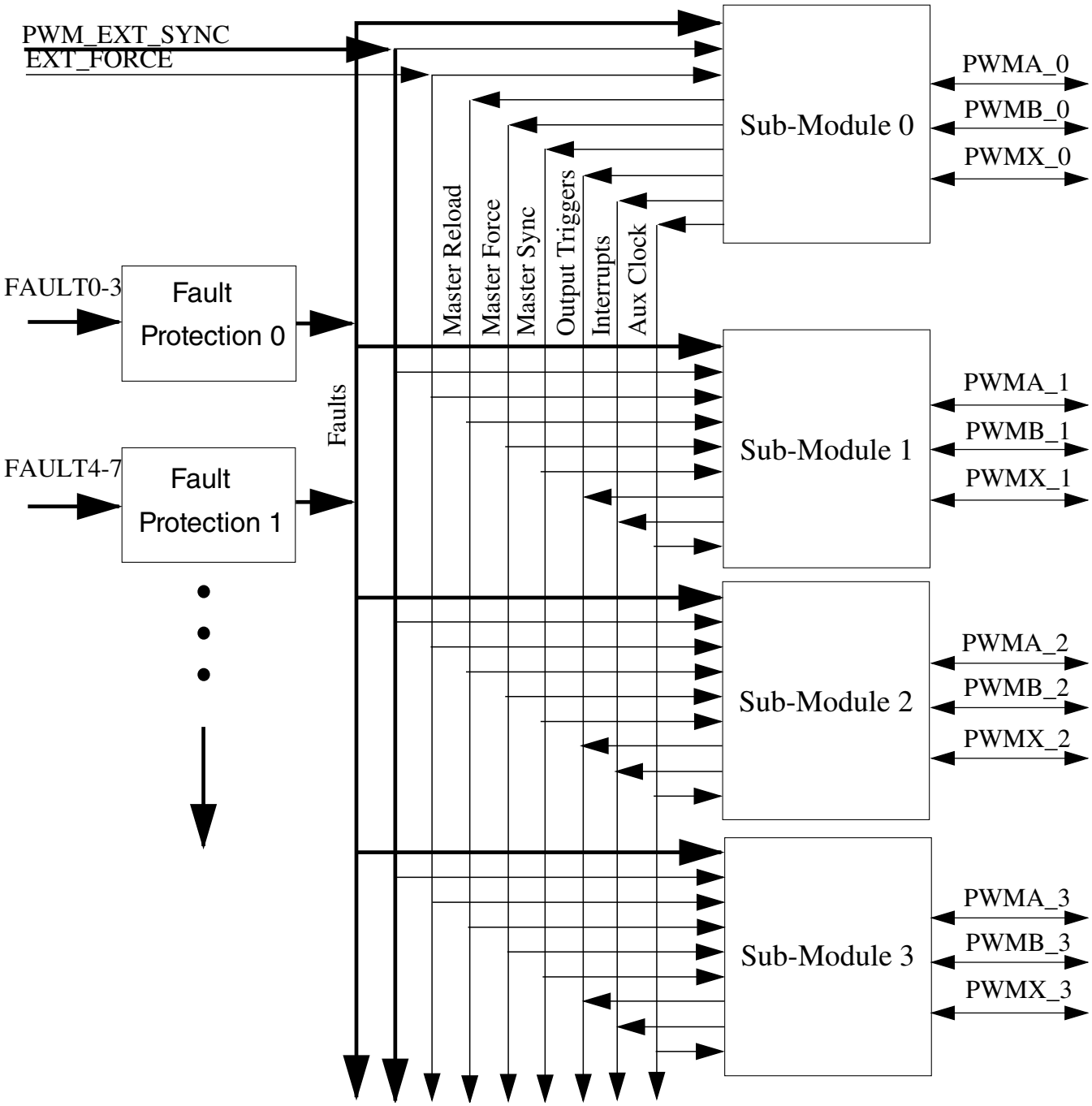


Figure 28-1. PWM Block Diagram

28.1.3.1 PWM Submodule

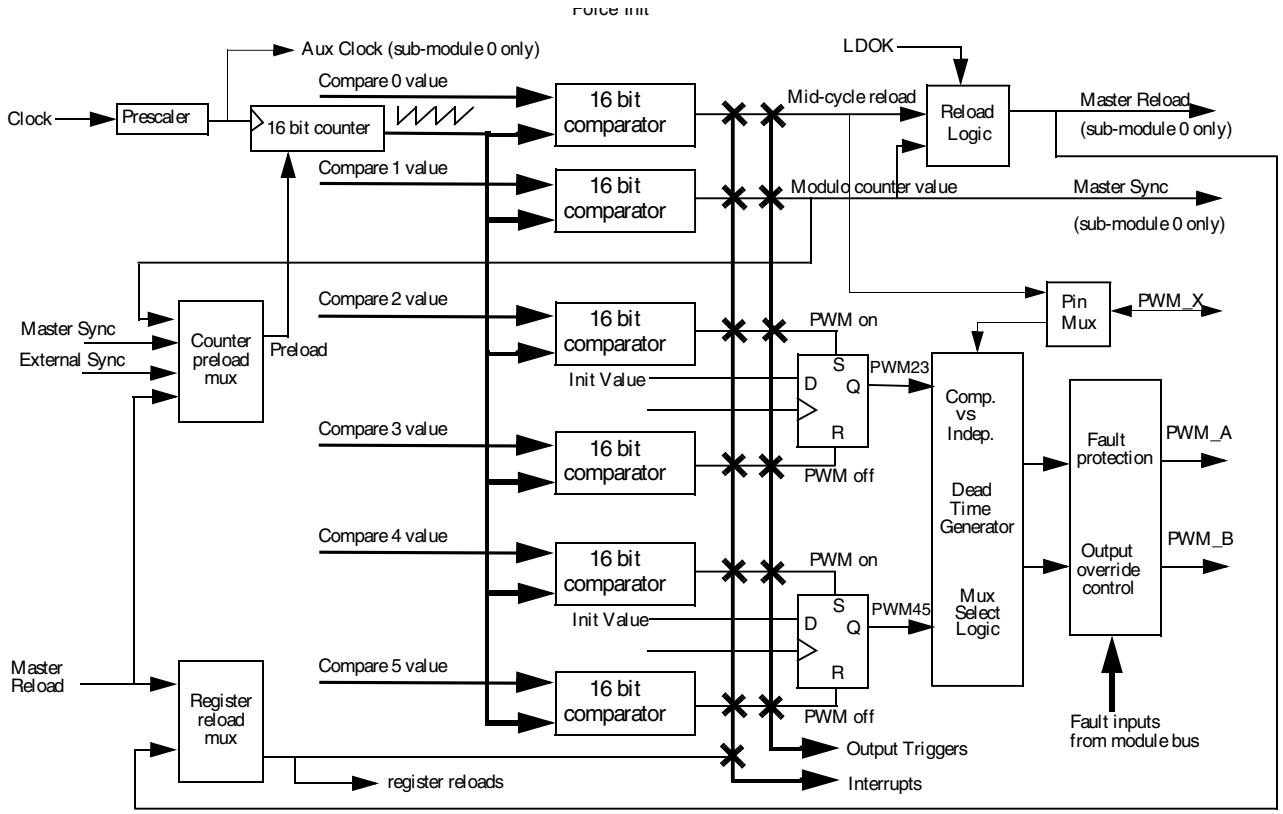


Figure 28-2. PWM Submodule Block Diagram

28.2 Signal Descriptions

The PWM has pins named PWM[n]_A, PWM[n]_B, PWM[n]_X, FAULT[n], PWM[n]_EXT_SYNC, EXT_FORCE, PWM[n]_EXTA, and PWM[n]_EXTB. The PWM also has an on-chip input called EXT_CLK and output signals called PWM[n]_OUT_TRIGx.

28.2.1 PWM[n]_A and PWM[n]_B - External PWM Output Pair

These pins are the output pins of the PWM channels. These pins can be independent PWM signals or a complementary pair. When not needed as an output, they can be used as inputs to the input capture circuitry.

28.2.2 PWM[n]_X - Auxiliary PWM Output signal

These pins are the auxiliary output pins of the PWM channels. They can be independent PWM signals. When not needed as an output, they can be used as inputs to the input capture circuitry or used to detect the polarity of the current flowing through the complementary circuit at deadtime correction.

28.2.3 FAULT[n] - Fault Inputs

These are input pins for disabling selected PWM outputs.

28.2.4 PWM[n]_EXT_SYNC - External Synchronization Signal

These input signals allow a source external to the PWM to initialize the PWM counter. In this manner, the PWM can be synchronized to external circuitry.

28.2.5 EXT_FORCE - External Output Force Signal

This input signal allows a source external to the PWM to force an update of the PWM outputs. In this manner, the PWM can be synchronized to external circuitry.

28.2.6 PWM[n]_EXTA and PWM[n]_EXTB - Alternate PWM Control Signals

These pins allow an alternate source to control the PWM_A and PWM_B outputs. Typically, either the PWM_EXTA or PWM_EXTB input (depending on the state of MCTRL[IPOL]) is used for the generation of a complementary pair. Typical control signals include ADC conversion high/low limits, TMR outputs, GPIO inputs, and comparator outputs.

28.2.7 PWM[n]_OUT_TRIG0 and PWM[n]_OUT_TRIG1 - Output Triggers

These outputs allow the PWM submodules to control timing of ADC conversions. See the description of the Output Trigger Control Register for information about how to enable these outputs and how the compare registers match up to the output triggers.

28.2.8 EXT_CLK - External Clock Signal

This signal allows a source external to the PWM (typically a timer or an off-chip source) to control the PWM clocking. In this manner, the PWM can be synchronized to the timer, or multiple chips can be synchronized to each other.

28.3 Memory Map and Registers

The address of a register is the sum of a base address and an address offset. The base address is defined at the core level, and the address offset is defined at the module level. The PWM module has a set of registers for each PWM submodule, for the configuration logic, and for each fault channel. While the registers are 16-bit wide, they can be accessed in pairs as 32-bit registers.

Submodule registers are repeated for each PWM submodule. To designate which submodule they are in, register names are prefixed with SM0, SM1, SM2, and SM3. The base address of submodule 0 is the same as the base address for the PWM module as a whole. The base address of submodule 1 is offset $30 from the base address for the PWM module as a whole. This $30 offset is based on the number of registers in a submodule. The base address of submodule 2 is equal to the base address of submodule 1 plus this same $30 offset. The pattern repeats for the base address of submodule 3.

The base address of the configuration registers is equal to the base address of the PWM module as a whole plus an offset of $C0.

Fault channel registers are repeated for each fault channel. To designate which fault channel they are in, register names are prefixed with F0 and F1. The base address of fault channel 0 is equal to the base address of the PWM module as a whole plus an offset of $C6. The base address of fault channel 1 is the base address of fault channel 0 + $4. This $4 offset is based on the number of registers in a fault channel. Each of the four fields in the fault channel registers corresponds to fault inputs 3-0.

PWMA memory map

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
E600	Counter Register (PWMA_SM0CNT)	16	R	0000h	28.3.1/636
E601	Initial Count Register (PWMA_SM0INIT)	16	R/W	0000h	28.3.2/637

Table continues on the next page...

PWMA memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
E602	Control 2 Register (PWMA_SM0CTRL2)	16	R/W	0000h	28.3.3/638
E603	Control Register (PWMA_SM0CTRL)	16	R/W	0400h	28.3.4/640
E605	Value Register 0 (PWMA_SM0VAL0)	16	R/W	0000h	28.3.5/642
E606	Fractional Value Register 1 (PWMA_SM0FRACVAL1)	16	R/W	0000h	28.3.6/643
E607	Value Register 1 (PWMA_SM0VAL1)	16	R/W	0000h	28.3.7/643
E608	Fractional Value Register 2 (PWMA_SM0FRACVAL2)	16	R/W	0000h	28.3.8/644
E609	Value Register 2 (PWMA_SM0VAL2)	16	R/W	0000h	28.3.9/644
E60A	Fractional Value Register 3 (PWMA_SM0FRACVAL3)	16	R/W	0000h	28.3.10/ 645
E60B	Value Register 3 (PWMA_SM0VAL3)	16	R/W	0000h	28.3.11/ 645
E60C	Fractional Value Register 4 (PWMA_SM0FRACVAL4)	16	R/W	0000h	28.3.12/ 646
E60D	Value Register 4 (PWMA_SM0VAL4)	16	R/W	0000h	28.3.13/ 646
E60E	Fractional Value Register 5 (PWMA_SM0FRACVAL5)	16	R/W	0000h	28.3.14/ 647
E60F	Value Register 5 (PWMA_SM0VAL5)	16	R/W	0000h	28.3.15/ 647
E610	Fractional Control Register (PWMA_SM0FRCTRL)	16	R/W	0000h	28.3.16/ 648
E611	Output Control Register (PWMA_SM0OCTRL)	16	R/W	0000h	28.3.17/ 649
E612	Status Register (PWMA_SM0STS)	16	w1c	0000h	28.3.18/ 651
E613	Interrupt Enable Register (PWMA_SM0INTEN)	16	R/W	0000h	28.3.19/ 653
E614	DMA Enable Register (PWMA_SM0DMAEN)	16	R/W	0000h	28.3.20/ 654
E615	Output Trigger Control Register (PWMA_SM0TCTRL)	16	R/W	0000h	28.3.21/ 656
E616	Fault Disable Mapping Register 0 (PWMA_SM0DISMAP0)	16	R/W	FFFFh	28.3.22/ 657
E617	Fault Disable Mapping Register 1 (PWMA_SM0DISMAP1)	16	R/W	FFFFh	28.3.23/ 657
E618	Deadtime Count Register 0 (PWMA_SM0DTCNT0)	16	R/W	07FFh	28.3.24/ 658
E619	Deadtime Count Register 1 (PWMA_SM0DTCNT1)	16	R/W	07FFh	28.3.25/ 659
E61A	Capture Control A Register (PWMA_SM0CAPTCTRLA)	16	R/W	0000h	28.3.26/ 659
E61B	Capture Compare A Register (PWMA_SM0CAPTCOMPA)	16	R/W	0000h	28.3.27/ 661

Table continues on the next page...

PWMA memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
E61C	Capture Control B Register (PWMA_SM0CAPTCTRLB)	16	R/W	0000h	28.3.28/ 662
E61D	Capture Compare B Register (PWMA_SM0CAPTCOMPB)	16	R/W	0000h	28.3.29/ 663
E61E	Capture Control X Register (PWMA_SM0CAPTCTRLX)	16	R/W	0000h	28.3.30/ 664
E61F	Capture Compare X Register (PWMA_SM0CAPTCOMPX)	16	R/W	0000h	28.3.31/ 666
E620	Capture Value 0 Register (PWMA_SM0CVAL0)	16	R	0000h	28.3.32/ 666
E621	Capture Value 0 Cycle Register (PWMA_SM0CVAL0CYC)	16	R	0000h	28.3.33/ 666
E622	Capture Value 1 Register (PWMA_SM0CVAL1)	16	R	0000h	28.3.34/ 667
E623	Capture Value 1 Cycle Register (PWMA_SM0CVAL1CYC)	16	R	0000h	28.3.35/ 667
E624	Capture Value 2 Register (PWMA_SM0CVAL2)	16	R	0000h	28.3.36/ 668
E625	Capture Value 2 Cycle Register (PWMA_SM0CVAL2CYC)	16	R	0000h	28.3.37/ 668
E626	Capture Value 3 Register (PWMA_SM0CVAL3)	16	R	0000h	28.3.38/ 668
E627	Capture Value 3 Cycle Register (PWMA_SM0CVAL3CYC)	16	R	0000h	28.3.39/ 669
E628	Capture Value 4 Register (PWMA_SM0CVAL4)	16	R	0000h	28.3.40/ 669
E629	Capture Value 4 Cycle Register (PWMA_SM0CVAL4CYC)	16	R	0000h	28.3.41/ 670
E62A	Capture Value 5 Register (PWMA_SM0CVAL5)	16	R	0000h	28.3.42/ 670
E62B	Capture Value 5 Cycle Register (PWMA_SM0CVAL5CYC)	16	R	0000h	28.3.43/ 670
E630	Counter Register (PWMA_SM1CNT)	16	R	0000h	28.3.1/636
E631	Initial Count Register (PWMA_SM1INIT)	16	R/W	0000h	28.3.2/637
E632	Control 2 Register (PWMA_SM1CTRL2)	16	R/W	0000h	28.3.3/638
E633	Control Register (PWMA_SM1CTRL)	16	R/W	0400h	28.3.4/640
E635	Value Register 0 (PWMA_SM1VAL0)	16	R/W	0000h	28.3.5/642
E636	Fractional Value Register 1 (PWMA_SM1FRACVAL1)	16	R/W	0000h	28.3.6/643
E637	Value Register 1 (PWMA_SM1VAL1)	16	R/W	0000h	28.3.7/643
E638	Fractional Value Register 2 (PWMA_SM1FRACVAL2)	16	R/W	0000h	28.3.8/644
E639	Value Register 2 (PWMA_SM1VAL2)	16	R/W	0000h	28.3.9/644
E63A	Fractional Value Register 3 (PWMA_SM1FRACVAL3)	16	R/W	0000h	28.3.10/ 645

Table continues on the next page...

PWMA memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
E63B	Value Register 3 (PWMA_SM1VAL3)	16	R/W	0000h	28.3.11/ 645
E63C	Fractional Value Register 4 (PWMA_SM1FRACVAL4)	16	R/W	0000h	28.3.12/ 646
E63D	Value Register 4 (PWMA_SM1VAL4)	16	R/W	0000h	28.3.13/ 646
E63E	Fractional Value Register 5 (PWMA_SM1FRACVAL5)	16	R/W	0000h	28.3.14/ 647
E63F	Value Register 5 (PWMA_SM1VAL5)	16	R/W	0000h	28.3.15/ 647
E640	Fractional Control Register (PWMA_SM1FRCTRL)	16	R/W	0000h	28.3.16/ 648
E641	Output Control Register (PWMA_SM1OCTRL)	16	R/W	0000h	28.3.17/ 649
E642	Status Register (PWMA_SM1STS)	16	w1c	0000h	28.3.18/ 651
E643	Interrupt Enable Register (PWMA_SM1INTEN)	16	R/W	0000h	28.3.19/ 653
E644	DMA Enable Register (PWMA_SM1DMAEN)	16	R/W	0000h	28.3.20/ 654
E645	Output Trigger Control Register (PWMA_SM1TCTRL)	16	R/W	0000h	28.3.21/ 656
E646	Fault Disable Mapping Register 0 (PWMA_SM1DISMAP0)	16	R/W	FFFFh	28.3.22/ 657
E647	Fault Disable Mapping Register 1 (PWMA_SM1DISMAP1)	16	R/W	FFFFh	28.3.23/ 657
E648	Deadtime Count Register 0 (PWMA_SM1DTCNT0)	16	R/W	07FFh	28.3.24/ 658
E649	Deadtime Count Register 1 (PWMA_SM1DTCNT1)	16	R/W	07FFh	28.3.25/ 659
E64A	Capture Control A Register (PWMA_SM1CAPTCTRLA)	16	R/W	0000h	28.3.26/ 659
E64B	Capture Compare A Register (PWMA_SM1CAPTCOMPA)	16	R/W	0000h	28.3.27/ 661
E64C	Capture Control B Register (PWMA_SM1CAPTCTRLB)	16	R/W	0000h	28.3.28/ 662
E64D	Capture Compare B Register (PWMA_SM1CAPTCOMPB)	16	R/W	0000h	28.3.29/ 663
E64E	Capture Control X Register (PWMA_SM1CAPTCTRLX)	16	R/W	0000h	28.3.30/ 664
E64F	Capture Compare X Register (PWMA_SM1CAPTCOMPX)	16	R/W	0000h	28.3.31/ 666
E650	Capture Value 0 Register (PWMA_SM1CVAL0)	16	R	0000h	28.3.32/ 666

Table continues on the next page...

PWMA memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
E651	Capture Value 0 Cycle Register (PWMA_SM1CVAL0CYC)	16	R	0000h	28.3.33/ 666
E652	Capture Value 1 Register (PWMA_SM1CVAL1)	16	R	0000h	28.3.34/ 667
E653	Capture Value 1 Cycle Register (PWMA_SM1CVAL1CYC)	16	R	0000h	28.3.35/ 667
E654	Capture Value 2 Register (PWMA_SM1CVAL2)	16	R	0000h	28.3.36/ 668
E655	Capture Value 2 Cycle Register (PWMA_SM1CVAL2CYC)	16	R	0000h	28.3.37/ 668
E656	Capture Value 3 Register (PWMA_SM1CVAL3)	16	R	0000h	28.3.38/ 668
E657	Capture Value 3 Cycle Register (PWMA_SM1CVAL3CYC)	16	R	0000h	28.3.39/ 669
E658	Capture Value 4 Register (PWMA_SM1CVAL4)	16	R	0000h	28.3.40/ 669
E659	Capture Value 4 Cycle Register (PWMA_SM1CVAL4CYC)	16	R	0000h	28.3.41/ 670
E65A	Capture Value 5 Register (PWMA_SM1CVAL5)	16	R	0000h	28.3.42/ 670
E65B	Capture Value 5 Cycle Register (PWMA_SM1CVAL5CYC)	16	R	0000h	28.3.43/ 670
E660	Counter Register (PWMA_SM2CNT)	16	R	0000h	28.3.1/636
E661	Initial Count Register (PWMA_SM2INIT)	16	R/W	0000h	28.3.2/637
E662	Control 2 Register (PWMA_SM2CTRL2)	16	R/W	0000h	28.3.3/638
E663	Control Register (PWMA_SM2CTRL)	16	R/W	0400h	28.3.4/640
E665	Value Register 0 (PWMA_SM2VAL0)	16	R/W	0000h	28.3.5/642
E666	Fractional Value Register 1 (PWMA_SM2FRACVAL1)	16	R/W	0000h	28.3.6/643
E667	Value Register 1 (PWMA_SM2VAL1)	16	R/W	0000h	28.3.7/643
E668	Fractional Value Register 2 (PWMA_SM2FRACVAL2)	16	R/W	0000h	28.3.8/644
E669	Value Register 2 (PWMA_SM2VAL2)	16	R/W	0000h	28.3.9/644
E66A	Fractional Value Register 3 (PWMA_SM2FRACVAL3)	16	R/W	0000h	28.3.10/ 645
E66B	Value Register 3 (PWMA_SM2VAL3)	16	R/W	0000h	28.3.11/ 645
E66C	Fractional Value Register 4 (PWMA_SM2FRACVAL4)	16	R/W	0000h	28.3.12/ 646
E66D	Value Register 4 (PWMA_SM2VAL4)	16	R/W	0000h	28.3.13/ 646
E66E	Fractional Value Register 5 (PWMA_SM2FRACVAL5)	16	R/W	0000h	28.3.14/ 647
E66F	Value Register 5 (PWMA_SM2VAL5)	16	R/W	0000h	28.3.15/ 647

Table continues on the next page...

PWMA memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
E670	Fractional Control Register (PWMA_SM2FRCTRL)	16	R/W	0000h	28.3.16/ 648
E671	Output Control Register (PWMA_SM2OCTRL)	16	R/W	0000h	28.3.17/ 649
E672	Status Register (PWMA_SM2STS)	16	w1c	0000h	28.3.18/ 651
E673	Interrupt Enable Register (PWMA_SM2INTEN)	16	R/W	0000h	28.3.19/ 653
E674	DMA Enable Register (PWMA_SM2DMAEN)	16	R/W	0000h	28.3.20/ 654
E675	Output Trigger Control Register (PWMA_SM2TCTRL)	16	R/W	0000h	28.3.21/ 656
E676	Fault Disable Mapping Register 0 (PWMA_SM2DISMAP0)	16	R/W	FFFFh	28.3.22/ 657
E677	Fault Disable Mapping Register 1 (PWMA_SM2DISMAP1)	16	R/W	FFFFh	28.3.23/ 657
E678	Deadtime Count Register 0 (PWMA_SM2DTCNT0)	16	R/W	07FFh	28.3.24/ 658
E679	Deadtime Count Register 1 (PWMA_SM2DTCNT1)	16	R/W	07FFh	28.3.25/ 659
E67A	Capture Control A Register (PWMA_SM2CAPTCTRLA)	16	R/W	0000h	28.3.26/ 659
E67B	Capture Compare A Register (PWMA_SM2CAPTCOMPA)	16	R/W	0000h	28.3.27/ 661
E67C	Capture Control B Register (PWMA_SM2CAPTCTRLB)	16	R/W	0000h	28.3.28/ 662
E67D	Capture Compare B Register (PWMA_SM2CAPTCOMPB)	16	R/W	0000h	28.3.29/ 663
E67E	Capture Control X Register (PWMA_SM2CAPTCTRLX)	16	R/W	0000h	28.3.30/ 664
E67F	Capture Compare X Register (PWMA_SM2CAPTCOMPX)	16	R/W	0000h	28.3.31/ 666
E680	Capture Value 0 Register (PWMA_SM2CVAL0)	16	R	0000h	28.3.32/ 666
E681	Capture Value 0 Cycle Register (PWMA_SM2CVAL0CYC)	16	R	0000h	28.3.33/ 666
E682	Capture Value 1 Register (PWMA_SM2CVAL1)	16	R	0000h	28.3.34/ 667
E683	Capture Value 1 Cycle Register (PWMA_SM2CVAL1CYC)	16	R	0000h	28.3.35/ 667
E684	Capture Value 2 Register (PWMA_SM2CVAL2)	16	R	0000h	28.3.36/ 668
E685	Capture Value 2 Cycle Register (PWMA_SM2CVAL2CYC)	16	R	0000h	28.3.37/ 668

Table continues on the next page...

PWMA memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
E686	Capture Value 3 Register (PWMA_SM2CVAL3)	16	R	0000h	28.3.38/ 668
E687	Capture Value 3 Cycle Register (PWMA_SM2CVAL3CYC)	16	R	0000h	28.3.39/ 669
E688	Capture Value 4 Register (PWMA_SM2CVAL4)	16	R	0000h	28.3.40/ 669
E689	Capture Value 4 Cycle Register (PWMA_SM2CVAL4CYC)	16	R	0000h	28.3.41/ 670
E68A	Capture Value 5 Register (PWMA_SM2CVAL5)	16	R	0000h	28.3.42/ 670
E68B	Capture Value 5 Cycle Register (PWMA_SM2CVAL5CYC)	16	R	0000h	28.3.43/ 670
E690	Counter Register (PWMA_SM3CNT)	16	R	0000h	28.3.1/636
E691	Initial Count Register (PWMA_SM3INIT)	16	R/W	0000h	28.3.2/637
E692	Control 2 Register (PWMA_SM3CTRL2)	16	R/W	0000h	28.3.3/638
E693	Control Register (PWMA_SM3CTRL)	16	R/W	0400h	28.3.4/640
E695	Value Register 0 (PWMA_SM3VAL0)	16	R/W	0000h	28.3.5/642
E696	Fractional Value Register 1 (PWMA_SM3FRACVAL1)	16	R/W	0000h	28.3.6/643
E697	Value Register 1 (PWMA_SM3VAL1)	16	R/W	0000h	28.3.7/643
E698	Fractional Value Register 2 (PWMA_SM3FRACVAL2)	16	R/W	0000h	28.3.8/644
E699	Value Register 2 (PWMA_SM3VAL2)	16	R/W	0000h	28.3.9/644
E69A	Fractional Value Register 3 (PWMA_SM3FRACVAL3)	16	R/W	0000h	28.3.10/ 645
E69B	Value Register 3 (PWMA_SM3VAL3)	16	R/W	0000h	28.3.11/ 645
E69C	Fractional Value Register 4 (PWMA_SM3FRACVAL4)	16	R/W	0000h	28.3.12/ 646
E69D	Value Register 4 (PWMA_SM3VAL4)	16	R/W	0000h	28.3.13/ 646
E69E	Fractional Value Register 5 (PWMA_SM3FRACVAL5)	16	R/W	0000h	28.3.14/ 647
E69F	Value Register 5 (PWMA_SM3VAL5)	16	R/W	0000h	28.3.15/ 647
E6A0	Fractional Control Register (PWMA_SM3FRCTRL)	16	R/W	0000h	28.3.16/ 648
E6A1	Output Control Register (PWMA_SM3OCTRL)	16	R/W	0000h	28.3.17/ 649
E6A2	Status Register (PWMA_SM3STS)	16	w1c	0000h	28.3.18/ 651
E6A3	Interrupt Enable Register (PWMA_SM3INTEN)	16	R/W	0000h	28.3.19/ 653
E6A4	DMA Enable Register (PWMA_SM3DMAEN)	16	R/W	0000h	28.3.20/ 654

Table continues on the next page...

PWMA memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
E6A5	Output Trigger Control Register (PWMA_SM3TCTRL)	16	R/W	0000h	28.3.21/ 656
E6A6	Fault Disable Mapping Register 0 (PWMA_SM3DISMAP0)	16	R/W	FFFFh	28.3.22/ 657
E6A7	Fault Disable Mapping Register 1 (PWMA_SM3DISMAP1)	16	R/W	FFFFh	28.3.23/ 657
E6A8	Deadtime Count Register 0 (PWMA_SM3DTCNT0)	16	R/W	07FFh	28.3.24/ 658
E6A9	Deadtime Count Register 1 (PWMA_SM3DTCNT1)	16	R/W	07FFh	28.3.25/ 659
E6AA	Capture Control A Register (PWMA_SM3CAPTCTRLA)	16	R/W	0000h	28.3.26/ 659
E6AB	Capture Compare A Register (PWMA_SM3CAPTCOMPA)	16	R/W	0000h	28.3.27/ 661
E6AC	Capture Control B Register (PWMA_SM3CAPTCTRLB)	16	R/W	0000h	28.3.28/ 662
E6AD	Capture Compare B Register (PWMA_SM3CAPTCOMPB)	16	R/W	0000h	28.3.29/ 663
E6AE	Capture Control X Register (PWMA_SM3CAPTCTRLX)	16	R/W	0000h	28.3.30/ 664
E6AF	Capture Compare X Register (PWMA_SM3CAPTCOMPX)	16	R/W	0000h	28.3.31/ 666
E6B0	Capture Value 0 Register (PWMA_SM3CVAL0)	16	R	0000h	28.3.32/ 666
E6B1	Capture Value 0 Cycle Register (PWMA_SM3CVAL0CYC)	16	R	0000h	28.3.33/ 666
E6B2	Capture Value 1 Register (PWMA_SM3CVAL1)	16	R	0000h	28.3.34/ 667
E6B3	Capture Value 1 Cycle Register (PWMA_SM3CVAL1CYC)	16	R	0000h	28.3.35/ 667
E6B4	Capture Value 2 Register (PWMA_SM3CVAL2)	16	R	0000h	28.3.36/ 668
E6B5	Capture Value 2 Cycle Register (PWMA_SM3CVAL2CYC)	16	R	0000h	28.3.37/ 668
E6B6	Capture Value 3 Register (PWMA_SM3CVAL3)	16	R	0000h	28.3.38/ 668
E6B7	Capture Value 3 Cycle Register (PWMA_SM3CVAL3CYC)	16	R	0000h	28.3.39/ 669
E6B8	Capture Value 4 Register (PWMA_SM3CVAL4)	16	R	0000h	28.3.40/ 669
E6B9	Capture Value 4 Cycle Register (PWMA_SM3CVAL4CYC)	16	R	0000h	28.3.41/ 670
E6BA	Capture Value 5 Register (PWMA_SM3CVAL5)	16	R	0000h	28.3.42/ 670

Table continues on the next page...

PWMA memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
E6BB	Capture Value 5 Cycle Register (PWMA_SM3CVAL5CYC)	16	R	0000h	28.3.43/ 670
E6C0	Output Enable Register (PWMA_OUTEN)	16	R/W	0000h	28.3.44/ 671
E6C1	Mask Register (PWMA_MASK)	16	R/W	0000h	28.3.45/ 672
E6C2	Software Controlled Output Register (PWMA_SWCOUT)	16	R/W	0000h	28.3.46/ 673
E6C3	PWM Source Select Register (PWMA_DTSRCSEL)	16	R/W	0000h	28.3.47/ 674
E6C4	Master Control Register (PWMA_MCTRL)	16	R/W	0000h	28.3.48/ 676
E6C5	Master Control 2 Register (PWMA_MCTRL2)	16	R/W	0000h	28.3.49/ 677
E6C6	Fault Control Register (PWMA_FCTRL0)	16	R/W	0000h	28.3.50/ 678
E6C7	Fault Status Register (PWMA_FSTS0)	16	R/W	0000h	28.3.51/ 679
E6C8	Fault Filter Register (PWMA_FFILT0)	16	R/W	0000h	28.3.52/ 680
E6C9	Fault Test Register (PWMA_FTST0)	16	R/W	0000h	28.3.53/ 681
E6CA	Fault Control Register (PWMA_FCTRL1)	16	R/W	0000h	28.3.50/ 678
E6CB	Fault Status Register (PWMA_FSTS1)	16	R/W	0000h	28.3.51/ 679
E6CC	Fault Filter Register (PWMA_FFILT1)	16	R/W	0000h	28.3.52/ 680
E6CD	Fault Test Register (PWMA_FTST1)	16	R/W	0000h	28.3.53/ 681

28.3.1 Counter Register (PWMA_SM*n*CNT)

This read-only register displays the state of the signed 16-bit submodule counter. This register is not byte accessible.

Address: E600h base + 0h offset + (48d × i), where i=0d to 3d

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	CNT															
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PWMA_SM*n*CNT field descriptions

Field	Description
CNT	Counter Register Bits

28.3.2 Initial Count Register (PWMA_SM*n*INIT)

The 16-bit signed value in this buffered, read/write register defines the initial count value for the PWM in PWM clock periods. This is the value loaded into the submodule counter when local sync, master sync, or master reload is asserted (based on the value of CTRL2[INIT_SEL]) or when CTRL2[FORCE] is asserted and force init is enabled. For PWM operation, the buffered contents of this register are loaded into the counter at the start of every PWM cycle. This register is not byte accessible.

NOTE

The INIT register is buffered. The value written does not take effect until MCTRL[LDOK] is set and the next PWM load cycle begins or CTRL[LDMOD] is set. This register cannot be written when MCTRL[LDOK] is set. Reading INIT reads the value in a buffer and not necessarily the value the PWM generator is currently using.

Address: E600h base + 1h offset + (48d × i), where i=0d to 3d

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read Write	INIT															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PWMA_SM*n*INIT field descriptions

Field	Description
INIT	Initial Count Register Bits

28.3.3 Control 2 Register (PWMA_SM*n*CTRL2)

Address: E600h base + 2h offset + (48d × i), where i=0d to 3d

Bit	15	14	13	12	11	10	9	8
Read Write	DBGEN	WAITEN	INDEP	PWM23_ INIT	PWM45_ INIT	PWMX_INIT	INIT_SEL	
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
Read	FRCEN	0	FORCE_SEL			RELOAD_ SEL	CLK_SEL	
Write		FORCE						
Reset	0	0	0	0	0	0	0	0

PWMA_SM*n*CTRL2 field descriptions

Field	Description
15 DBGEN	Debug Enable When set to one, the PWM will continue to run while the chip is in debug mode. If the device enters debug mode and this bit is zero, then the PWM outputs will be disabled until debug mode is exited. At that point the PWM pins will resume operation as programmed in the PWM registers. For certain types of motors (such as 3-phase AC), it is imperative that this bit be left in its default state (in which the PWM is disabled in debug mode). Failure to do so could result in damage to the motor or inverter. For other types of motors (example: DC motors), this bit might safely be set to one, enabling the PWM in debug mode. The key point is PWM parameter updates will not occur in debug mode. Any motors requiring such updates should be disabled during debug mode. If in doubt, leave this bit set to zero.
14 WAITEN	WAIT Enable When set to one, the PWM will continue to run while the chip is in WAIT mode. In this mode, the peripheral clock continues to run but the CPU clock does not. If the device enters WAIT mode and this bit is zero, then the PWM outputs will be disabled until WAIT mode is exited. At that point the PWM pins will resume operation as programmed in the PWM registers. For certain types of motors (such as 3-phase AC), it is imperative that this bit be left in its default state (in which the PWM is disabled in WAIT mode). Failure to do so could result in damage to the motor or inverter. For other types of motors (example: DC motors), this bit might safely be set to one, enabling the PWM in WAIT mode. The key point is PWM parameter updates will not occur in this mode. Any motors requiring such updates should be disabled during WAIT mode. If in doubt, leave this bit set to zero.
13 INDEP	Independent or Complementary Pair Operation This bit determines if the PWM_A and PWM_B channels will be independent PWMs or a complementary PWM pair. 0 PWM_A and PWM_B form a complementary PWM pair. 1 PWM_A and PWM_B outputs are independent PWMs.
12 PWM23_INIT	PWM23 Initial Value This read/write bit determines the initial value for PWM23 and the value to which it is forced when FORCE_INIT is asserted.
11 PWM45_INIT	PWM45 Initial Value

Table continues on the next page...

PWMA_SM*n*CTRL2 field descriptions (continued)

Field	Description
	This read/write bit determines the initial value for PWM45 and the value to which it is forced when FORCE_INIT is asserted.
10 PWMX_INIT	PWM_X Initial Value This read/write bit determines the initial value for PWM_X and the value to which it is forced when FORCE_INIT is asserted.
9–8 INIT_SEL	Initialization Control Select These read/write bits control the source of the INIT signal which goes to the counter. 00 Local sync (PWM_X) causes initialization. 01 Master reload from submodule 0 causes initialization. This setting should not be used in submodule 0 as it will force the INIT signal to logic 0. The submodule counter will only reinitialize when a master reload occurs. 10 Master sync from submodule 0 causes initialization. This setting should not be used in submodule 0 as it will force the INIT signal to logic 0. 11 EXT_SYNC causes initialization.
7 FRCEN	Force Initialization Enable This bit allows the CTRL2[FORCE] signal to initialize the counter without regard to the signal selected by CTRL2[INIT_SEL]. This is a software controlled initialization. 0 Initialization from a FORCE_OUT event is disabled. 1 Initialization from a FORCE_OUT event is enabled.
6 FORCE	Force Initialization If CTRL2[FORCE_SEL] is set to 000, writing a 1 to this bit results in a FORCE_OUT event. This causes the following actions to be taken: • The PWM_A and PWM_B output pins will assume values based on DTSRCSEL[SMxSEL23] and DTSRCSEL[SMxSEL45]. • If CTRL2[FRCEN] is set, the counter value will be initialized with the INIT register value.
5–3 FORCE_SEL	This read/write bit determines the source of the FORCE OUTPUT signal for this submodule. 000 The local force signal, CTRL2[FORCE], from this submodule is used to force updates. 001 The master force signal from submodule 0 is used to force updates. This setting should not be used in submodule 0 as it will hold the FORCE OUTPUT signal to logic 0. 010 The local reload signal from this submodule is used to force updates without regard to the state of LDOK. 011 The master reload signal from submodule0 is used to force updates if LDOK is set. This setting should not be used in submodule0 as it will hold the FORCE OUTPUT signal to logic 0. 100 The local sync signal from this submodule is used to force updates. 101 The master sync signal from submodule0 is used to force updates. This setting should not be used in submodule0 as it will hold the FORCE OUTPUT signal to logic 0. 110 The external force signal, EXT_FORCE, from outside the PWM module causes updates. 111 The external sync signal, EXT_SYNC, from outside the PWM module causes updates.
2 RELOAD_SEL	Reload Source Select This read/write bit determines the source of the RELOAD signal for this submodule. When this bit is set, MCTRL[LDOK[0]] for submodule 0 should be used since the local MCTRL[LDOK] will be ignored.

Table continues on the next page...

PWMA_SM*n*CTRL2 field descriptions (continued)

Field	Description
	0 The local RELOAD signal is used to reload registers. 1 The master RELOAD signal (from submodule 0) is used to reload registers. This setting should not be used in submodule 0 as it will force the RELOAD signal to logic 0.
CLK_SEL	Clock Source Select These read/write bits determine the source of the clock signal for this submodule. 00 The IPBus clock is used as the clock for the local prescaler and counter. 01 EXT_CLK is used as the clock for the local prescaler and counter. 10 Submodule 0's clock (AUX_CLK) is used as the source clock for the local prescaler and counter. This setting should not be used in submodule 0 as it will force the clock to logic 0. 11 reserved

28.3.4 Control Register (PWMA_SM*n*CTRL)

Address: E600h base + 3h offset + (48d × i), where i=0d to 3d

Bit	15	14	13	12	11	10	9	8
Read	LDFQ				HALF	FULL	DT	
Write								
Reset	0	0	0	0	0	1	0	0

Bit	7	6	5	4	3	2	1	0
Read	0	PRSC			0	LDMOD	DBLX	DBLEN
Write								
Reset	0	0	0	0	0	0	0	0

PWMA_SM*n*CTRL field descriptions

Field	Description
15–12 LDFQ	These buffered read/write bits select the PWM load frequency. Reset clears LDFQ, selecting loading every PWM opportunity. A PWM opportunity is determined by HALF and FULL. **NOTE:** LDFQ takes effect when the current load cycle is complete, regardless of the state of MCTRL[LDOK]. Reading LDFQ reads the buffered values and not necessarily the values currently in effect. 0000 Every PWM opportunity 0001 Every 2 PWM opportunities 0010 Every 3 PWM opportunities 0011 Every 4 PWM opportunities 0100 Every 5 PWM opportunities 0101 Every 6 PWM opportunities 0110 Every 7 PWM opportunities 0111 Every 8 PWM opportunities 1000 Every 9 PWM opportunities 1001 Every 10 PWM opportunities

Table continues on the next page...

PWMA_SM*n*CTRL field descriptions (continued)

Field	Description
	1010 Every 11 PWM opportunities 1011 Every 12 PWM opportunities 1100 Every 13 PWM opportunities 1101 Every 14 PWM opportunities 1110 Every 15 PWM opportunities 1111 Every 16 PWM opportunities
11 HALF	Half Cycle Reload This read/write bit enables half-cycle reloads. A half cycle is defined by when the submodule counter matches the VAL0 register and does not have to be half way through the PWM cycle. 0 Half-cycle reloads disabled. 1 Half-cycle reloads enabled.
10 FULL	Full Cycle Reload This read/write bit enables full-cycle reloads. A full cycle is defined by when the submodule counter matches the VAL1 register. Either CTRL[HALF] or CTRL[FULL] must be set in order to move the buffered data into the registers used by the PWM generators or CTRL[LDMOD] must be set. If both CTRL[HALF] and CTRL[FULL] are set, then reloads can occur twice per cycle. 0 Full-cycle reloads disabled. 1 Full-cycle reloads enabled.
9–8 DT	Deadtime These read only bits reflect the sampled values of the PWM_X input at the end of each deadtime. Sampling occurs at the end of deadtime 0 for DT[0] and the end of deadtime 1 for DT[1]. Reset clears these bits.
7 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
6–4 PRSC	Prescaler These buffered read/write bits select the divide ratio of the PWM clock frequency selected by CTRL2[CLK_SEL]. **NOTE:** Reading CTRL[PRSC] reads the buffered values and not necessarily the values currently in effect. CTRL[PRSC] takes effect at the beginning of the next PWM cycle and only when the load okay bit, MCTRL[LDOK], is set or CTRL[LDMOD] is set. This field cannot be written when MCTRL[LDOK] is set. 000 PWM clock frequency = f_{clk} 001 PWM clock frequency = $f_{clk}/2$ 010 PWM clock frequency = $f_{clk}/4$ 011 PWM clock frequency = $f_{clk}/8$ 100 PWM clock frequency = $f_{clk}/16$ 101 PWM clock frequency = $f_{clk}/32$ 110 PWM clock frequency = $f_{clk}/64$ 111 PWM clock frequency = $f_{clk}/128$
3 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
2 LDMOD	Load Mode Select This read/write bit selects the timing of loading the buffered registers for this submodule.

Table continues on the next page...

PWMA_SM*n*CTRL field descriptions (continued)

Field	Description
	0 Buffered registers of this submodule are loaded and take effect at the next PWM reload if MCTRL[LDOK] is set. 1 Buffered registers of this submodule are loaded and take effect immediately upon MCTRL[LDOK] being set. In this case it is not necessary to set CTRL[FULL] or CTRL[HALF].
1 DBLX	PWMX Double Switching Enable This read/write bit enables the double switching behavior on PWMX. When this bit is set, the PWMX output shall be the exclusive OR combination of PWMA and PWMB prior to polarity and masking considerations. 0 PWMX double pulse disabled. 1 PWMX double pulse enabled.
0 DBLEN	Double Switching Enable This read/write bit enables the double switching PWM behavior(See Double Switching PWMs). Double switching is not compatible with fractional PWM clock generation. Make sure this bit is clear when setting FRCTRL[FRAC23_EN], FRCTRL[FRAC45_EN], or FRCTRL[FRAC1_EN]. 0 Double switching disabled. 1 Double switching enabled.

28.3.5 Value Register 0 (PWMA_SM*n*VAL0)

Address: E600h base + 5h offset + (48d × i), where i=0d to 3d

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read Write	VAL0															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PWMA_SM*n*VAL0 field descriptions

Field	Description
VAL0	Value Register 0 The 16-bit signed value in this buffered, read/write register defines the mid-cycle reload point for the PWM in PWM clock periods. This value also defines when the PWM_X signal is set and the local sync signal is reset. This register is not byte accessible. **NOTE:** The VAL0 register is buffered. The value written does not take effect until MCTRL[LDOK] is set and the next PWM load cycle begins or CTRL[LDMOD] is set. VAL0 cannot be written when MCTRL[LDOK] is set. Reading VAL0 reads the value in a buffer. It is not necessarily the value the PWM generator is currently using.

28.3.6 Fractional Value Register 1 (PWMA_SM*n*FRACVAL1)

Address: E600h base + 6h offset + (48d × i), where i=0d to 3d

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	FRACVAL1					0										
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PWMA_SM*n*FRACVAL1 field descriptions

Field	Description
15–11 FRACVAL1	Fractional Value 1 Register These bits act as a fractional addition to the value in the VAL1 register which controls the PWM period width. The PWM period is computed in terms of IPBus clock cycles. This fractional portion is accumulated at the end of every cycle until an additional whole IPBus cycle is reached. At this time the value being used for VAL1 is temporarily incremented and the PWM cycle is extended by one clock period to compensate for the accumulated fractional values. **NOTE:** The FRACVAL1 register is buffered. The value written does not take effect until MCTRL[LDOK] is set and the next PWM load cycle begins or CTRL[LDMOD] is set. FRACVAL1 cannot be written when MCTRL[LDOK] is set. Reading FRACVAL1 reads the value in a buffer and not necessarily the value the PWM generator is currently using.
Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

28.3.7 Value Register 1 (PWMA_SM*n*VAL1)

Address: E600h base + 7h offset + (48d × i), where i=0d to 3d

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	VAL1															
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PWMA_SM*n*VAL1 field descriptions

Field	Description
VAL1	Value Register 1 The 16-bit signed value written to this buffered, read/write register defines the modulo count value (maximum count) for the submodule counter. Upon reaching this count value, the counter reloads itself with the contents of the INIT register and asserts the local sync signal while resetting PWM_X. This register is not byte accessible. **NOTE:** The VAL1 register is buffered. The value written does not take effect until MCTRL[LDOK] is set and the next PWM load cycle begins or CTRL[LDMOD] is set. VAL1 cannot be written when MCTRL[LDOK] is set. Reading VAL1 reads the value in a buffer. It is not necessarily the value the PWM generator is currently using. **NOTE:** When using FRACVAL1, limit the maximum value of VAL1 to 0xFFFE for unsigned applications or to 0x7FFE for signed applications, to avoid counter rollovers caused by accumulating the fractional period defined by FRACVAL1.

PWMA_SM*n*VAL1 field descriptions (continued)

Field	Description
	NOTE: If the VAL1 register defines the timer period (Local Sync is selected as the counter initialization signal), a 100% duty cycle cannot be achieved on the PWMX output. After the count reaches VAL1, the PWMX output is low for a minimum of one count every cycle. When the Master Sync signal (only originated by the Local Sync from sub-module 0) is used to control the timer period, the VAL1 register can be free for other functions such as PWM generation without the duty cycle limitation.

28.3.8 Fractional Value Register 2 (PWMA_SM*n*FRACVAL2)

Address: E600h base + 8h offset + (48d × i), where i=0d to 3d

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	FRACVAL2					0										
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PWMA_SM*n*FRACVAL2 field descriptions

Field	Description
15–11 FRACVAL2	Fractional Value 2 These bits act as a fractional addition to the value in the VAL2 register which controls the PWM_A turn on timing. It is also used to control the fractional addition to the turn off delay of PWM_B when MCTRL[IPOLx]=0 in complementary mode, CTRL2[INDEP]=0. **NOTE:** The FRACVAL2 register is buffered. The value written does not take effect until MCTRL[LDOK] is set and the next PWM load cycle begins or CTRL[LDMOD] is set. FRACVAL2 cannot be written when MCTRL[LDOK] is set. Reading FRACVAL2 reads the value in a buffer and not necessarily the value the PWM generator is currently using. **NOTE:** FRCTRL[FRAC23_EN] should be set to 0 when the values of VAL2 and VAL3 cause the high or low time of the PWM output to be 3 cycles or less.
Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

28.3.9 Value Register 2 (PWMA_SM*n*VAL2)

Address: E600h base + 9h offset + (48d × i), where i=0d to 3d

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	VAL2															
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PWMA_SM*n*VAL2 field descriptions

Field	Description
VAL2	Value Register 2

PWMA_SM*n*VAL2 field descriptions (continued)

Field	Description
	The 16-bit signed value in this buffered, read/write register defines the count value to set PWM23 high. This register is not byte accessible. **NOTE:** The VAL2 register is buffered. The value written does not take effect until MCTRL[LDOK] is set and the next PWM load cycle begins or CTRL[LDMOD] is set. VAL2 cannot be written when MCTRL[LDOK] is set. Reading VAL2 reads the value in a buffer and not necessarily the value the PWM generator is currently using.

28.3.10 Fractional Value Register 3 (PWMA_SM*n*FRACVAL3)

Address: E600h base + Ah offset + (48d × i), where i=0d to 3d

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	FRACVAL3					0										
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PWMA_SM*n*FRACVAL3 field descriptions

Field	Description
15–11 FRACVAL3	Fractional Value 3 These bits act as a fractional addition to the value in the VAL3 register which controls the PWM_A turn off timing. It is also used to control the fractional addition to the turn on delay of PWM_B when MCTRL[IPOLx]=0 in complementary mode, CTRL2[INDEP]=0. **NOTE:** The FRACVAL3 register is buffered. The value written does not take effect until MCTRL[LDOK] is set and the next PWM load cycle begins or CTRL[LDMOD] is set. FRACVAL3 cannot be written when MCTRL[LDOK] is set. Reading FRACVAL3 reads the value in a buffer and not necessarily the value the PWM generator is currently using. **NOTE:** FRCTRL[FRAC23_EN] should be set to 0 when the values of VAL2 and VAL3 cause the high or low time of the PWM output to be 3 cycles or less.
Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

28.3.11 Value Register 3 (PWMA_SM*n*VAL3)

Address: E600h base + Bh offset + (48d × i), where i=0d to 3d

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	VAL3															
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PWMA_SM*n*VAL3 field descriptions

Field	Description
VAL3	Value Register 3

PWMA_SM*n*VAL3 field descriptions (continued)

Field	Description
	The 16-bit signed value in this buffered, read/write register defines the count value to set PWM23 low. This register is not byte accessible. **NOTE:** The VAL3 register is buffered. The value written does not take effect until MCTRL[LDOK] is set and the next PWM load cycle begins or CTRL[LDMOD] is set. VAL3 cannot be written when MCTRL[LDOK] is set. Reading VAL3 reads the value in a buffer and not necessarily the value the PWM generator is currently using.

28.3.12 Fractional Value Register 4 (PWMA_SM*n*FRACVAL4)

Address: E600h base + Ch offset + (48d × i), where i=0d to 3d

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	FRACVAL4					0										
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PWMA_SM*n*FRACVAL4 field descriptions

Field	Description
15–11 FRACVAL4	Fractional Value 4 These bits act as a fractional addition to the value in the VAL4 register which controls the PWM_B turn on timing. It is also used to control the fractional addition to the turn off delay of PWM_A when MCTRL[IPOLx]=1 in complementary mode, CTRL2[INDEP]=0. **NOTE:** The FRACVAL4 register is buffered. The value written does not take effect until MCTRL[LDOK] is set and the next PWM load cycle begins or CTRL[LDMOD] is set. FRACVAL4 cannot be written when MCTRL[LDOK] is set. Reading FRACVAL4 reads the value in a buffer and not necessarily the value the PWM generator is currently using. **NOTE:** FRCTRL[FRAC45_EN] should be set to 0 when the values of VAL4 and VAL5 cause the high or low time of the PWM output to be 3 cycles or less.
Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

28.3.13 Value Register 4 (PWMA_SM*n*VAL4)

Address: E600h base + Dh offset + (48d × i), where i=0d to 3d

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	VAL4															
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PWMA_SM*n*VAL4 field descriptions

Field	Description
VAL4	Value Register 4

PWMA_SM*n*VAL4 field descriptions (continued)

Field	Description
	The 16-bit signed value in this buffered, read/write register defines the count value to set PWM45 high. This register is not byte accessible. **NOTE:** The VAL4 register is buffered. The value written does not take effect until MCTRL[LDOK] is set and the next PWM load cycle begins or CTRL[LDMOD] is set. VAL4 cannot be written when MCTRL[LDOK] is set. Reading VAL4 reads the value in a buffer and not necessarily the value the PWM generator is currently using.

28.3.14 Fractional Value Register 5 (PWMA_SM*n*FRACVAL5)

Address: E600h base + Eh offset + (48d × i), where i=0d to 3d

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	FRACVAL5					0										
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PWMA_SM*n*FRACVAL5 field descriptions

Field	Description
15–11 FRACVAL5	Fractional Value 5 These bits act as a fractional addition to the value in the VAL5 register which controls the PWM_B turn off timing. It is also used to control the fractional addition to the turn on delay of PWM_A when MCTRL[IPOLx]=1 in complementary mode, CTRL2[INDEP]=0. **NOTE:** The FRACVAL5 register is buffered. The value written does not take effect until MCTRL[LDOK] is set and the next PWM load cycle begins or CTRL[LDMOD] is set. FRACVAL5 cannot be written when MCTRL[LDOK] is set. Reading FRACVAL5 reads the value in a buffer and not necessarily the value the PWM generator is currently using. **NOTE:** FRCTRL[FRAC45_EN] should be set to 0 when the values of VAL4 and VAL5 cause the high or low time of the PWM output to be 3 cycles or less.
Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

28.3.15 Value Register 5 (PWMA_SM*n*VAL5)

Address: E600h base + Fh offset + (48d × i), where i=0d to 3d

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	VAL5															
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PWMA_SM*n*VAL5 field descriptions

Field	Description
VAL5	Value Register 5

PWMA_SM*n*VAL5 field descriptions (continued)

Field	Description
	The 16-bit signed value in this buffered, read/write register defines the count value to set PWM45 low. This register is not byte accessible. **NOTE:** The VAL5 register is buffered. The value written does not take effect until MCTRL[LDOK] is set and the next PWM load cycle begins or CTRL[LDMOD] is set. VAL5 cannot be written when MCTRL[LDOK] is set. Reading VAL5 reads the value in a buffer and not necessarily the value the PWM generator is currently using.

28.3.16 Fractional Control Register (PWMA_SM*n*FRCTRL)

Address: E600h base + 10h offset + (48d × i), where i=0d to 3d

Bit	15	14	13	12	11	10	9	8
Read	TEST	0						FRAC_PU
Write								
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
Read	0			FRAC45_EN	0	FRAC23_EN	FRAC1_EN	0
Write								
Reset	0	0	0	0	0	0	0	0

PWMA_SM*n*FRCTRL field descriptions

Field	Description
15 TEST	Test Status Bit This is a read only test bit for factory use. This bit will reset to 0 but may be either 0 or 1 during PWM operation.
14–9 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
8 FRAC_PU	Fractional Delay Circuit Power Up This bit is used to power up the fractional delay analog block. The fractional delay block takes 25 us to power up after the first FRAC_PU bit in any submodule is set. The fractional delay block only powers down when the FRAC_PU bits in all submodules are 0. The fractional delay logic can only be used when the IPBus clock is running at 100 MHz. When turned off, fractional placement is disabled. 0 Turn off fractional delay logic. 1 Power up fractional delay logic.
7–5 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
4 FRAC45_EN	Fractional Cycle Placement Enable for PWM_B This bit is used to enable the fractional cycle edge placement of PWM_B using the FRACVAL4 and FRACVAL5 registers. When disabled, the fractional cycle edge placement of PWM_B is bypassed.

Table continues on the next page...

PWMA_SM*n*FRCTRL field descriptions (continued)

Field	Description
	NOTE: The FRAC45_EN bit is buffered. The value written does not take effect until MCTRL[LDOK] is set and the next PWM load cycle begins or CTRL[LDMOD] is set. FRAC45_EN cannot be written when MCTRL[LDOK] is set. Reading FRAC45_EN reads the value in a buffer and not necessarily the value the PWM generator is currently using. 0 Disable fractional cycle placement for PWM_B. 1 Enable fractional cycle placement for PWM_B.
3 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
2 FRAC23_EN	Fractional Cycle Placement Enable for PWM_A This bit is used to enable the fractional cycle edge placement of PWM_A using the FRACVAL2 and FRACVAL3 registers. When disabled, the fractional cycle edge placement of PWM_A is bypassed. **NOTE:** The FRAC23_EN bit is buffered. The value written does not take effect until MCTRL[LDOK] is set and the next PWM load cycle begins or CTRL[LDMOD] is set. FRAC23_EN cannot be written when MCTRL[LDOK] is set. Reading FRAC23_EN reads the value in a buffer and not necessarily the value the PWM generator is currently using. 0 Disable fractional cycle placement for PWM_A. 1 Enable fractional cycle placement for PWM_A.
1 FRAC1_EN	Fractional Cycle PWM Period Enable This bit is used to enable the fractional cycle length of the PWM period using the FRACVAL1 register. When disabled, the fractional cycle length of the PWM period is bypassed. **NOTE:** The FRAC1_EN bit is buffered. The value written does not take effect until MCTRL[LDOK] is set and the next PWM load cycle begins or CTRL[LDMOD] is set. FRAC1_EN cannot be written when MCTRL[LDOK] is set. Reading FRAC1_EN reads the value in a buffer and not necessarily the value the PWM generator is currently using. 0 Disable fractional cycle length for the PWM period. 1 Enable fractional cycle length for the PWM period.
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

28.3.17 Output Control Register (PWMA_SM*n*OCTRL)

Address: E600h base + 11h offset + (48d × i), where i=0d to 3d

Bit	15	14	13	12	11	10	9	8
Read	PWMA_IN	PWMB_IN	PWMX_IN	0		POLA	POLB	POLX
Write								
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
Read	0		PWMAFS		PWMBFS		PWMXFS	
Write								
Reset	0	0	0	0	0	0	0	0

PWMA_SM*n*OCTRL field descriptions

Field	Description
15 PWMA_IN	PWM_A Input This read only bit shows the logic value currently being driven into the PWM_A input. The bit's reset state is undefined.
14 PWMB_IN	PWM_B Input This read only bit shows the logic value currently being driven into the PWM_B input. The bit's reset state is undefined.
13 PWMX_IN	PWM_X Input This read only bit shows the logic value currently being driven into the PWM_X input. The bit's reset state is undefined.
12–11 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
10 POLA	PWM_A Output Polarity This bit inverts the PWM_A output polarity. 0 PWM_A output not inverted. A high level on the PWM_A pin represents the "on" or "active" state. 1 PWM_A output inverted. A low level on the PWM_A pin represents the "on" or "active" state.
9 POLB	PWM_B Output Polarity This bit inverts the PWM_B output polarity. 0 PWM_B output not inverted. A high level on the PWM_B pin represents the "on" or "active" state. 1 PWM_B output inverted. A low level on the PWM_B pin represents the "on" or "active" state.
8 POLX	PWM_X Output Polarity This bit inverts the PWM_X output polarity. 0 PWM_X output not inverted. A high level on the PWM_X pin represents the "on" or "active" state. 1 PWM_X output inverted. A low level on the PWM_X pin represents the "on" or "active" state.
7–6 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
5–4 PWMAFS	PWM_A Fault State These bits determine the fault state for the PWM_A output during fault conditions and STOP mode. It may also define the output state during WAIT and DEBUG modes depending on the settings of CTRL2[WAITEN] and CTRL2[DBGEN]. 00 Output is forced to logic 0 state prior to consideration of output polarity control. 01 Output is forced to logic 1 state prior to consideration of output polarity control. 10 Output is tristated. 11 Output is tristated.
3–2 PWMBFS	PWM_B Fault State These bits determine the fault state for the PWM_B output during fault conditions and STOP mode. It may also define the output state during WAIT and DEBUG modes depending on the settings of CTRL2[WAITEN] and CTRL2[DBGEN]. 00 Output is forced to logic 0 state prior to consideration of output polarity control. 01 Output is forced to logic 1 state prior to consideration of output polarity control.

Table continues on the next page...

PWMA_SM*n*OCTRL field descriptions (continued)

Field	Description
	10 Output is tristated. 11 Output is tristated.
PWMXFS	PWM_X Fault State These bits determine the fault state for the PWM_X output during fault conditions and STOP mode. It may also define the output state during WAIT and DEBUG modes depending on the settings of CTRL2[WAITEN] and CTRL2[DBGEN]. 00 Output is forced to logic 0 state prior to consideration of output polarity control. 01 Output is forced to logic 1 state prior to consideration of output polarity control. 10 Output is tristated. 11 Output is tristated.

28.3.18 Status Register (PWMA_SM*n*STS)

Address: E600h base + 12h offset + (48d × i), where i=0d to 3d

Bit	15	14	13	12	11	10	9	8
Read	0	RUF	REF	RF	CFA1	CFA0	CFB1	CFB0
Write			w1c	w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
Read	CFX1	CFX0	CMPF					
Write	w1c	w1c	w1c					
Reset	0	0	0	0	0	0	0	0

PWMA_SM*n*STS field descriptions

Field	Description
15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
14 RUF	Registers Updated Flag This read-only flag is set when one of the INIT, VALx,FRACVALx, or CTRL[PRSC] registers has been written, which indicates potentially non-coherent data in the set of double buffered registers. Clear this bit by a proper reload sequence consisting of a reload signal while MCTRL[LDOK] = 1. Reset clears this bit. 0 No register update has occurred since last reload. 1 At least one of the double buffered registers has been updated since the last reload.
13 REF	Reload Error Flag This read/write flag is set when a reload cycle occurs while MCTRL[LDOK] is 0 and the double buffered registers are in a non-coherent state (STS[RUF] = 1). Clear this bit by writing a logic one to this location. Reset clears this bit.

Table continues on the next page...

PWMA_SM*n*STS field descriptions (continued)

Field	Description
	0 No reload error occurred. 1 Reload signal occurred with non-coherent data and MCTRL[LDOK] = 0.
12 RF	Reload Flag This read/write flag is set at the beginning of every reload cycle regardless of the state of MCTRL[LDOK]. Clear this bit by writing a logic one to this location when DMAEN[VALDE] is clear (non-DMA mode). This flag can also be cleared by the DMA done signal when DMAEN[VALDE] is set (DMA mode). Reset clears this bit. 0 No new reload cycle since last STS[RF] clearing 1 New reload cycle since last STS[RF] clearing
11 CFA1	Capture Flag A1 This bit is set when a capture event occurs on the Capture A1 circuit. This bit is cleared by writing a one to this bit position if DMAEN[CA1DE] is clear (non-DMA mode) or by the DMA done signal if DMAEN[CA1DE] is set (DMA mode). Reset clears this bit.
10 CFA0	Capture Flag A0 This bit is set when a capture event occurs on the Capture A0 circuit. This bit is cleared by writing a one to this bit position if DMAEN[CA0DE] is clear (non-DMA mode) or by the DMA done signal if DMAEN[CA0DE] is set (DMA mode). Reset clears this bit.
9 CFB1	Capture Flag B1 This bit is set when a capture event occurs on the Capture B1 circuit. This bit is cleared by writing a one to this bit position if DMAEN[CB1DE] is clear (non-DMA mode) or by the DMA done signal if DMAEN[CB1DE] is set (DMA mode). Reset clears this bit.
8 CFB0	Capture Flag B0 This bit is set when a capture event occurs on the Capture B0 circuit. This bit is cleared by writing a one to this bit position if DMAEN[CB0DE] is clear (non-DMA mode) or by the DMA done signal if DMAEN[CB0DE] is set (DMA mode). Reset clears this bit.
7 CFX1	Capture Flag X1 This bit is set when a capture event occurs on the Capture X1 circuit. This bit is cleared by writing a one to this bit position if DMAEN[CX1DE] is clear (non-DMA mode) or by the DMA done signal if DMAEN[CX1DE] is set (DMA mode). Reset clears this bit.
6 CFX0	Capture Flag X0 This bit is set when a capture event occurs on the Capture X0 circuit. This bit is cleared by writing a one to this bit position if DMAEN[CX0DE] is clear (non-DMA mode) or by the DMA done signal if DMAEN[CX0DE] is set (DMA mode). Reset clears this bit.
CMPF	Compare Flags These bits are set when the submodule counter value matches the value of one of the VALx registers. Clear these bits by writing a 1 to a bit position. 0 No compare event has occurred for a particular VALx value. 1 A compare event has occurred for a particular VALx value.

28.3.19 Interrupt Enable Register (PWMA_SM*n*INTEN)

Address: E600h base + 13h offset + (48d × i), where i=0d to 3d

Bit	15	14	13	12	11	10	9	8
Read	0		REIE	RIE	CA1IE	CA0IE	CB1IE	CB0IE
Write								
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
Read	CX1IE	CX0IE	CMPIE					
Write								
Reset	0	0	0	0	0	0	0	0

PWMA_SM*n*INTEN field descriptions

Field	Description
15–14 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
13 REIE	Reload Error Interrupt Enable This read/write bit enables the reload error flag, STS[REF], to generate CPU interrupt requests. Reset clears this bit. 0 STS[REF] CPU interrupt requests disabled 1 STS[REF] CPU interrupt requests enabled
12 RIE	Reload Interrupt Enable This read/write bit enables the reload flag, STS[RF], to generate CPU interrupt requests. Reset clears this bit. 0 STS[RF] CPU interrupt requests disabled 1 STS[RF] CPU interrupt requests enabled
11 CA1IE	Capture A 1 Interrupt Enable This bit allows the STS[CFA1] flag to create an interrupt request to the CPU. Do not set both this bit and DMAEN[CA1DE]. 0 Interrupt request disabled for STS[CFA1]. 1 Interrupt request enabled for STS[CFA1].
10 CA0IE	Capture A 0 Interrupt Enable This bit allows the STS[CFA0] flag to create an interrupt request to the CPU. Do not set both this bit and DMAEN[CA0DE]. 0 Interrupt request disabled for STS[CFA0]. 1 Interrupt request enabled for STS[CFA0].
9 CB1IE	Capture B 1 Interrupt Enable This bit allows the STS[CFB1] flag to create an interrupt request to the CPU. Do not set both this bit and DMAEN[CB1DE]. 0 Interrupt request disabled for STS[CFB1]. 1 Interrupt request enabled for STS[CFB1].

Table continues on the next page...

PWMA_SM*n*INTEN field descriptions (continued)

Field	Description
8 CB0IE	Capture B 0 Interrupt Enable This bit allows the STS[CFB0] flag to create an interrupt request to the CPU. Do not set both this bit and DMAEN[CB0DE]. 0 Interrupt request disabled for STS[CFB0]. 1 Interrupt request enabled for STS[CFB0].
7 CX1IE	Capture X 1 Interrupt Enable This bit allows the STS[CFX1] flag to create an interrupt request to the CPU. Do not set both this bit and DMAEN[CX1DE]. 0 Interrupt request disabled for STS[CFX1]. 1 Interrupt request enabled for STS[CFX1].
6 CX0IE	Capture X 0 Interrupt Enable This bit allows the STS[CFX0] flag to create an interrupt request to the CPU. Do not set both this bit and DMAEN[CX0DE]. 0 Interrupt request disabled for STS[CFX0]. 1 Interrupt request enabled for STS[CFX0].
CMPIE	Compare Interrupt Enables These bits enable the STS[CMPF] flags to cause a compare interrupt request to the CPU. 0 The corresponding STS[CMPF] bit will not cause an interrupt request. 1 The corresponding STS[CMPF] bit will cause an interrupt request.

28.3.20 DMA Enable Register (PWMA_SM*n*DMAEN)

Address: E600h base + 14h offset + (48d × i), where i=0d to 3d

Bit	15	14	13	12	11	10	9	8
Read	0						VALDE	FAND
Write								
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
Read Write	CAPTDE		CA1DE	CA0DE	CB1DE	CB0DE	CX1DE	CX0DE
Reset	0	0	0	0	0	0	0	0

PWMA_SM*n*DMAEN field descriptions

Field	Description
15–10 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
9 VALDE	Value Registers DMA Enable

Table continues on the next page...

PWMA_SM*n*DMAEN field descriptions (continued)

Field	Description
	This read/write bit enables DMA write requests for the VALx and FRACVALx registers when STS[RF] is set. Reset clears this bit. 0 DMA write requests disabled 1 DMA write requests for the VALx and FRACVALx registers enabled
8 FAND	FIFO Watermark AND Control This read/write bit works in conjunction with the DMAEN[CAPTDE] field when it is set to watermark mode (DMAEN[CAPTDE] = 01). While DMAEN[CAxDE], DMAEN[CBxDE], and DMAEN[CXxDE] determine which FIFO watermarks the DMA read request is sensitive to, this bit determines if the selected watermarks are AND'ed together or OR'ed together in order to create the request. 0 Selected FIFO watermarks are OR'ed together. 1 Selected FIFO watermarks are AND'ed together.
7–6 CAPTDE	Capture DMA Enable Source Select These read/write bits select the source of enabling the DMA read requests for the capture FIFOs. Reset clears these bits. 00 Read DMA requests disabled. 01 Exceeding a FIFO watermark sets the DMA read request. This requires at least one of DMAEN[CA1DE], DMAEN[CA0DE], DMAEN[CB1DE], DMAEN[CB0DE], DMAEN[CX1DE], or DMAEN[CX0DE] to also be set in order to determine to which watermark(s) the DMA request is sensitive. 10 A local sync (VAL1 matches counter) sets the read DMA request. 11 A local reload (STS[RF] being set) sets the read DMA request.
5 CA1DE	Capture A1 FIFO DMA Enable This read/write bit enables DMA read requests for the Capture A1 FIFO data when STS[CFA1] is set. Reset clears this bit. Do not set both this bit and INTEN[CA1IE].
4 CA0DE	Capture A0 FIFO DMA Enable This read/write bit enables DMA read requests for the Capture A0 FIFO data when STS[CFA0] is set. Reset clears this bit. Do not set both this bit and INTEN[CA0IE].
3 CB1DE	Capture B1 FIFO DMA Enable This read/write bit enables DMA read requests for the Capture B1 FIFO data when STS[CFB1] is set. Reset clears this bit. Do not set both this bit and INTEN[CB1IE].
2 CB0DE	Capture B0 FIFO DMA Enable This read/write bit enables DMA read requests for the Capture B0 FIFO data when STS[CFB0] is set. Reset clears this bit. Do not set both this bit and INTEN[CB0IE].
1 CX1DE	Capture X1 FIFO DMA Enable This read/write bit enables DMA read requests for the Capture X1 FIFO data when STS[CFX1] is set. Reset clears this bit. Do not set both this bit and INTEN[CX1IE].
0 CX0DE	Capture X0 FIFO DMA Enable This read/write bit enables DMA read requests for the Capture X0 FIFO data when STS[CFX0] is set. Reset clears this bit. Do not set both this bit and INTEN[CX0IE].

28.3.21 Output Trigger Control Register (PWMA_SM*n*TCTRL)

Address: E600h base + 15h offset + (48d × i), where i=0d to 3d

Bit	15	14	13	12	11	10	9	8
Read	0	0	0					
Write	PWAOT0	PWBOT1						
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
Read	0		OUT_TRIG_EN					
Write								
Reset	0	0	0	0	0	0	0	0

PWMA_SM*n*TCTRL field descriptions

Field	Description
15 PWAOT0	Output Trigger 0 Source Select This bit selects which signal to bring out on the PWM's PWM_OUT_TRIG0 port. The output trigger port is often connected to routing logic on the chip. This control bit allows the PWMA output signal to be driven onto the output trigger port so it can be sent to the chip routing logic. 0 Route the PWM_OUT_TRIG0 signal to PWM_OUT_TRIG0 port. 1 Route the PWMA output to the PWM_OUT_TRIG0 port.
14 PWBOT1	Output Trigger 1 Source Select This bit selects which signal to bring out on the PWM's PWM_OUT_TRIG1 port. The output trigger port is often connected to routing logic on the chip. This control bit allows the PWMB output signal to be driven onto the output trigger port so it can be sent to the chip routing logic. 0 Route the PWM_OUT_TRIG1 signal to PWM_OUT_TRIG1 port. 1 Route the PWMB output to the PWM_OUT_TRIG1 port.
13–6 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
OUT_TRIG_EN	Output Trigger Enables These bits enable the generation of PWM_OUT_TRIG0 and PWM_OUT_TRIG1 outputs based on the counter value matching the value in one or more of the VAL0-5 registers. VAL0, VAL2, and VAL4 are used to generate PWM_OUT_TRIG0, and VAL1, VAL3, and VAL5 are used to generate PWM_OUT_TRIG1. The PWM_OUT_TRIGx signals are only asserted as long as the counter value matches the VALx value; therefore, up to six triggers can be generated (three each on PWM_OUT_TRIG0 and PWM_OUT_TRIG1) per PWM cycle per submodule. **NOTE:** Due to delays in creating the PWM outputs, the output trigger signals will lead the PWM output edges by 2-3 clock cycles depending on the fractional cycle value being used. 0 PWM_OUT_TRIGx will not set when the counter value matches the VALx value. 1 PWM_OUT_TRIGx will set when the counter value matches the VALx value.

28.3.22 Fault Disable Mapping Register 0 (PWMA_SM*n*DISMAP0)

This register determines which PWM pins are disabled by the fault protection inputs. Reset sets all of the bits in the fault disable mapping register.

Address: E600h base + 16h offset + (48d × i), where i=0d to 3d

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	1				DIS0X				DIS0B				DIS0A			
Write																
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

PWMA_SM*n*DISMAP0 field descriptions

Field	Description
15–12 Reserved	This field is reserved. This read-only field is reserved and always has the value 1.
11–8 DIS0X	PWM_X Fault Disable Mask 0 Each of the four bits of this read/write field is one-to-one associated with the four FAULTx inputs of fault channel 0. The PWM_X output is turned off if there is a logic 1 on a FAULTx input and a 1 in the corresponding bit of this field. A reset sets all bits in this field.
7–4 DIS0B	PWM_B Fault Disable Mask 0 Each of the four bits of this read/write field is one-to-one associated with the four FAULTx inputs of fault channel 0. The PWM_B output is turned off if there is a logic 1 on a FAULTx input and a 1 in the corresponding bit of this field. A reset sets all bits in this field.
DIS0A	PWM_A Fault Disable Mask 0 Each of the four bits of this read/write field is one-to-one associated with the four FAULTx inputs of fault channel 0. The PWM_A output is turned off if there is a logic 1 on a FAULTx input and a 1 in the corresponding bit of this field. A reset sets all bits in this field.

28.3.23 Fault Disable Mapping Register 1 (PWMA_SM*n*DISMAP1)

This register determines which PWM pins are disabled by the fault protection inputs. Reset sets all of the bits in the fault disable mapping register.

Address: E600h base + 17h offset + (48d × i), where i=0d to 3d

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	1				DIS1X				DIS1B				DIS1A			
Write																
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

PWMA_SM*n*DISMAP1 field descriptions

Field	Description
15–12 Reserved	This field is reserved. This read-only field is reserved and always has the value 1.
11–8 DIS1X	PWM_X Fault Disable Mask 1 Each of the four bits of this read/write field is one-to-one associated with the four FAULTx inputs of fault channel 1. The PWM_X output is turned off if there is a logic 1 on a FAULTx input and a 1 in the corresponding bit of this field. A reset sets all bits in this field.
7–4 DIS1B	PWM_B Fault Disable Mask 1 Each of the four bits of this read/write field is one-to-one associated with the four FAULTx inputs of fault channel 1. The PWM_B output is turned off if there is a logic 1 on a FAULTx input and a 1 in the corresponding bit of this field. A reset sets all bits in this field.
DIS1A	PWM_A Fault Disable Mask 1 Each of the four bits of this read/write field is one-to-one associated with the four FAULTx inputs of fault channel 1. The PWM_A output is turned off if there is a logic 1 on a FAULTx input and a 1 in the corresponding bit of this field. A reset sets all bits in this field.

28.3.24 Deadtime Count Register 0 (PWMA_SM*n*DTCNT0)

Deadtime operation applies only to complementary channel operation. The values written to the DTCNTx registers are in terms of IPBus clock cycles regardless of the setting of CTRL[PRSC] and/or CTRL2[CLK_SEL]. Reset sets the deadtime count registers to a default value of 0x07FF, selecting a deadtime of 2047 IPBus clock cycles. The DTCNTx registers are not byte accessible.

Address: E600h base + 18h offset + (48d × i), where i=0d to 3d

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	0					DTCNT0										
Write																
Reset	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1

PWMA_SM*n*DTCNT0 field descriptions

Field	Description
15–11 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
DTCNT0	Deadtime Count Register 0 The DTCNT0 field is used to control the deadtime during 0 to 1 transitions of the PWM_A output (assuming normal polarity).

28.3.25 Deadtime Count Register 1 (PWMA_SM*n*DTCNT1)

Deadtime operation applies only to complementary channel operation. The values written to the DTCNTx registers are in terms of IPBus clock cycles regardless of the setting of CTRL[PRSC] and/or CTRL2[CLK_SEL]. Reset sets the deadtime count registers to a default value of 0x07FF, selecting a deadtime of 2047 IPBus clock cycles. The DTCNTx registers are not byte accessible.

Address: E600h base + 19h offset + (48d × i), where i=0d to 3d

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	0					DTCNT1										
Write																
Reset	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1

PWMA_SM*n*DTCNT1 field descriptions

Field	Description
15–11 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
DTCNT1	Deadtime Count Register 1 The DTCNT1 field is used to control the deadtime during 0 to 1 transitions of the complementary PWM_B output.

28.3.26 Capture Control A Register (PWMA_SM*n*CAPTCTRLA)

Address: E600h base + 1Ah offset + (48d × i), where i=0d to 3d

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	CA1CNT			CA0CNT			CFAWM		EDGCNTA_EN	INP_SELA	EDGA1		EDGA0		ONESHOTA	ARMA
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PWMA_SM*n*CAPTCTRLA field descriptions

Field	Description
15–13 CA1CNT	Capture A1 FIFO Word Count

Table continues on the next page...

PWMA_SM*n*CAPTCTRLA field descriptions (continued)

Field	Description
	This field reflects the number of words in the Capture A1 FIFO. (FIFO depth is 1)
12–10 CA0CNT	Capture A0 FIFO Word Count This field reflects the number of words in the Capture A0 FIFO. (FIFO depth is 1)
9–8 CFAWM	Capture A FIFOs Water Mark This field represents the water mark level for capture A FIFOs. The capture flags, STS[CFA1] and STS[CFA0], are not set until the word count of the corresponding FIFO is greater than this water mark level. (FIFO depth is 1)
7 EDGCNTA_EN	Edge Counter A Enable This bit enables the edge counter which counts rising and falling edges on the PWM_A input signal. 0 Edge counter disabled and held in reset 1 Edge counter enabled
6 INP_SELA	Input Select A This bit selects between the raw PWM_A input signal and the output of the edge counter/compare circuitry as the source for the input capture circuit. 0 Raw PWM_A input signal selected as source. 1 Output of edge counter/compare selected as source. **NOTE:** When this bitfield is set to 1, the internal edge counter is enabled and the rising and/or falling edges specified by the CAPTCTRLA[EDGA0] and CAPTCTRLA[EDGA1] fields are ignored. The software must still place a value other than 00 in either or both of the CAPTCTLRA[EDGA0] and/or CAPTCTRLA[EDGA1] fields in order to enable one or both of the capture registers.
5–4 EDGA1	Edge A 1 These bits control the input capture 1 circuitry by determining which input edges cause a capture event. 00 Disabled 01 Capture falling edges 10 Capture rising edges 11 Capture any edge
3–2 EDGA0	Edge A 0 These bits control the input capture 0 circuitry by determining which input edges cause a capture event. 00 Disabled 01 Capture falling edges 10 Capture rising edges 11 Capture any edge
1 ONESHOTA	One Shot Mode A This bit selects between free running and one shot mode for the input capture circuitry. 0 Free running mode is selected. If both capture circuits are enabled, then capture circuit 0 is armed first after CAPTCTRLA[ARMA] is set. Once a capture occurs, capture circuit 0 is disarmed and capture circuit 1 is armed. After capture circuit 1 performs a capture, it is disarmed and capture circuit 0 is re-armed. The process continues indefinitely.

Table continues on the next page...

PWMA_SM*n*CAPTCTRLA field descriptions (continued)

Field	Description
	If only one of the capture circuits is enabled, then captures continue indefinitely on the enabled capture circuit. 1 One shot mode is selected. If both capture circuits are enabled, then capture circuit 0 is armed first after CAPTCTRLA[ARMA] is set. Once a capture occurs, capture circuit 0 is disarmed and capture circuit 1 is armed. After capture circuit 1 performs a capture, it is disarmed and CAPTCTRLA[ARMA] is cleared. No further captures will be performed until CAPTCTRLA[ARMA] is set again. If only one of the capture circuits is enabled, then a single capture will occur on the enabled capture circuit and CAPTCTRLA[ARMA] is then cleared.
0 ARMA	Arm A Setting this bit high starts the input capture process. This bit can be cleared at any time to disable input capture operation. This bit is self-cleared when in one shot mode and one or more of the enabled capture circuits has had a capture event. 0 Input capture operation is disabled. 1 Input capture operation as specified by CAPTCTRLA[EDGAx] is enabled.

28.3.27 Capture Compare A Register (PWMA_SM*n*CAPTCOMPA)

Address: E600h base + 1Bh offset + (48d × i), where i=0d to 3d

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	EDGCNTA								EDGCMPA							
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PWMA_SM*n*CAPTCOMPA field descriptions

Field	Description
15–8 EDGCNTA	Edge Counter A This read-only field contains the edge counter value for the PWM_A input capture circuitry.
EDGCMPA	Edge Compare A This read/write field is the compare value associated with the edge counter for the PWM_A input capture circuitry.

28.3.28 Capture Control B Register (PWMA_SM*n*CAPTCTRLB)

Address: E600h base + 1Ch offset + (48d × i), where i=0d to 3d

Bit	15	14	13	12	11	10	9	8
Read	CB1CNT				CB0CNT		CFBWM	
Write								
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
Read	EDGCNTB_EN	INP_SELB	EDGB1		EDGB0		ONESHOTB	ARMB
Write								
Reset	0	0	0	0	0	0	0	0

PWMA_SM*n*CAPTCTRLB field descriptions

Field	Description
15–13 CB1CNT	Capture B1 FIFO Word Count This field reflects the number of words in the Capture B1 FIFO. (FIFO depth is 1)
12–10 CB0CNT	Capture B0 FIFO Word Count This field reflects the number of words in the Capture B0 FIFO. (FIFO depth is 1)
9–8 CFBWM	Capture B FIFOs Water Mark This field represents the water mark level for capture B FIFOs. The capture flags, STS[CFB1] and STS[CFB0], won't be set until the word count of the corresponding FIFO is greater than this water mark level. (FIFO depth is 1)
7 EDGCNTB_EN	Edge Counter B Enable This bit enables the edge counter which counts rising and falling edges on the PWM_B input signal. 0 Edge counter disabled and held in reset 1 Edge counter enabled
6 INP_SELB	Input Select B This bit selects between the raw PWM_B input signal and the output of the edge counter/compare circuitry as the source for the input capture circuit. 0 Raw PWM_B input signal selected as source. 1 Output of edge counter/compare selected as source. **NOTE:** When this bitfield is set to 1, the internal edge counter is enabled and the rising and/or falling edges specified by the CAPTCTRLB[EDGB0] and CAPTCTRLB[EDGB1] fields are ignored. The software must still place a value other than 00 in either or both of the CAPTCTLRB[EDGB0] and/or CAPTCTRLB[EDGB1] fields in order to enable one or both of the capture registers.
5–4 EDGB1	Edge B 1 These bits control the input capture 1 circuitry by determining which input edges cause a capture event. 00 Disabled

Table continues on the next page...

PWMA_SM*n*CAPTCTRLB field descriptions (continued)

Field	Description
	01 Capture falling edges 10 Capture rising edges 11 Capture any edge
3–2 EDGB0	Edge B 0 These bits control the input capture 0 circuitry by determining which input edges cause a capture event. 00 Disabled 01 Capture falling edges 10 Capture rising edges 11 Capture any edge
1 ONESHOTB	One Shot Mode B This bit selects between free running and one shot mode for the input capture circuitry. 0 Free running mode is selected. If both capture circuits are enabled, then capture circuit 0 is armed first after CAPTCTRLB[ARMB] is set. Once a capture occurs, capture circuit 0 is disarmed and capture circuit 1 is armed. After capture circuit 1 performs a capture, it is disarmed and capture circuit 0 is re-armed. The process continues indefinitely. If only one of the capture circuits is enabled, then captures continue indefinitely on the enabled capture circuit. 1 One shot mode is selected. If both capture circuits are enabled, then capture circuit 0 is armed first after CAPTCTRLB[ARMB] is set. Once a capture occurs, capture circuit 0 is disarmed and capture circuit 1 is armed. After capture circuit 1 performs a capture, it is disarmed and CAPTCTRLB[ARMB] is cleared. No further captures will be performed until CAPTCTRLB[ARMB] is set again. If only one of the capture circuits is enabled, then a single capture will occur on the enabled capture circuit and CAPTCTRLB[ARMB] is then cleared.
0 ARMB	Arm B Setting this bit high starts the input capture process. This bit can be cleared at any time to disable input capture operation. This bit is self-cleared when in one shot mode and one or more of the enabled capture circuits has had a capture event. 0 Input capture operation is disabled. 1 Input capture operation as specified by CAPTCTRLB[EDGBx] is enabled.

28.3.29 Capture Compare B Register (PWMA_SM*n*CAPTCOMPB)

Address: E600h base + 1Dh offset + (48d × i), where i=0d to 3d

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	EDGCNTB								EDGCMPB							
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PWMA_SM*n*CAPTCOMPB field descriptions

Field	Description
15–8 EDGCNTB	Edge Counter B This read-only field contains the edge counter value for the PWM_B input capture circuitry.
EDGCMPB	Edge Compare B This read/write field is the compare value associated with the edge counter for the PWM_B input capture circuitry.

28.3.30 Capture Control X Register (PWMA_SM*n*CAPTCTRLX)

Address: E600h base + 1Eh offset + (48d × i), where i=0d to 3d

Bit	15	14	13	12	11	10	9	8
Read	CX1CNT			CX0CNT			CFXWM	
Write								
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
Read	EDGCNTX_EN	INP_SELX	EDGX1		EDGX0		ONESHOTX	ARMX
Write								
Reset	0	0	0	0	0	0	0	0

PWMA_SM*n*CAPTCTRLX field descriptions

Field	Description
15–13 CX1CNT	Capture X1 FIFO Word Count This field reflects the number of words in the Capture X1 FIFO. (FIFO depth is 1)
12–10 CX0CNT	Capture X0 FIFO Word Count This field reflects the number of words in the Capture X0 FIFO. (FIFO depth is 1)
9–8 CFXWM	Capture X FIFOs Water Mark This field represents the water mark level for capture X FIFOs. The capture flags, STS[CFX1] and STS[CFX0], won't be set until the word count of the corresponding FIFO is greater than this water mark level. (FIFO depth is 1)
7 EDGCNTX_EN	Edge Counter X Enable This bit enables the edge counter which counts rising and falling edges on the PWM_X input signal. 0 Edge counter disabled and held in reset 1 Edge counter enabled
6 INP_SELX	Input Select X This bit selects between the raw PWM_X input signal and the output of the edge counter/compare circuitry as the source for the input capture circuit.

Table continues on the next page...

PWMA_SM*n*CAPTCTRLX field descriptions (continued)

Field	Description
	0 Raw PWM_X input signal selected as source. 1 Output of edge counter/compare selected as source. **NOTE:** When this bitfield is set to 1, the internal edge counter is enabled and the rising and/or falling edges specified by the CAPTCTRLX[EDGX0] and CAPTCTRLX[EDGX1] fields are ignored. The software must still place a value other than 00 in either or both of the CAPTCTLRX[EDGX0] and/or CAPTCTRLX[EDGX1] fields in order to enable one or both of the capture registers.
5–4 EDGX1	Edge X 1 These bits control the input capture 1 circuitry by determining which input edges cause a capture event. 00 Disabled 01 Capture falling edges 10 Capture rising edges 11 Capture any edge
3–2 EDGX0	Edge X 0 These bits control the input capture 0 circuitry by determining which input edges cause a capture event. 00 Disabled 01 Capture falling edges 10 Capture rising edges 11 Capture any edge
1 ONESHOTX	One Shot Mode Aux This bit selects between free running and one shot mode for the input capture circuitry. 0 Free running mode is selected. If both capture circuits are enabled, then capture circuit 0 is armed first after the ARMX bit is set. Once a capture occurs, capture circuit 0 is disarmed and capture circuit 1 is armed. After capture circuit 1 performs a capture, it is disarmed and capture circuit 0 is re-armed. The process continues indefinitely. If only one of the capture circuits is enabled, then captures continue indefinitely on the enabled capture circuit. 1 One shot mode is selected. If both capture circuits are enabled, then capture circuit 0 is armed first after the ARMX bit is set. Once a capture occurs, capture circuit 0 is disarmed and capture circuit 1 is armed. After capture circuit 1 performs a capture, it is disarmed and the ARMX bit is cleared. No further captures will be performed until the ARMX bit is set again. If only one of the capture circuits is enabled, then a single capture will occur on the enabled capture circuit and the ARMX bit is then cleared.
0 ARMX	Arm X Setting this bit high starts the input capture process. This bit can be cleared at any time to disable input capture operation. This bit is self-cleared when in one shot mode and one or more of the enabled capture circuits has had a capture event. 0 Input capture operation is disabled. 1 Input capture operation as specified by CAPTCTRLX[EDGXx] is enabled.

28.3.31 Capture Compare X Register (PWMA_SM*n*CAPTCOMPX)

Address: E600h base + 1Fh offset + (48d × i), where i=0d to 3d

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	EDGCNTX								EDGCMPX							
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PWMA_SM*n*CAPTCOMPX field descriptions

Field	Description
15–8 EDGCNTX	Edge Counter X This read-only field contains the edge counter value for the PWM_X input capture circuitry.
EDGCMPX	Edge Compare X This read/write field is the compare value associated with the edge counter for the PWM_X input capture circuitry.

28.3.32 Capture Value 0 Register (PWMA_SM*n*CVAL0)

Address: E600h base + 20h offset + (48d × i), where i=0d to 3d

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	CAPTVAL0															
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PWMA_SM*n*CVAL0 field descriptions

Field	Description
CAPTVAL0	This read-only register stores the value captured from the submodule counter. Exactly when this capture occurs is defined by CAPTCTRLX[EDGX0]. Each capture will increase the value of CAPTCTRLX[CX0CNT] by 1 until the maximum value is reached. Each read of this register will decrease the value of CAPTCTRLX[CX0CNT] by 1 until 0 is reached. This register is not byte accessible.

28.3.33 Capture Value 0 Cycle Register (PWMA_SM*n*CVAL0CYC)

Address: E600h base + 21h offset + (48d × i), where i=0d to 3d

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	0												CVAL0CYC			
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PWMA_SM*n*CVAL0CYC field descriptions

Field	Description
15–4 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
CVAL0CYC	This read-only register stores the cycle number corresponding to the value captured in CVAL0. This register is incremented each time the counter is loaded with the INIT value at the end of a PWM modulo cycle.

28.3.34 Capture Value 1 Register (PWMA_SM*n*CVAL1)

Address: E600h base + 22h offset + (48d × i), where i=0d to 3d

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	CAPTVAL1															
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PWMA_SM*n*CVAL1 field descriptions

Field	Description
CAPTVAL1	This read-only register stores the value captured from the submodule counter. Exactly when this capture occurs is defined by CAPTCTRLX[EDGX1]. Each capture increases the value of CAPTCTRLX[CX1CNT] by 1 until the maximum value is reached. Each read of this register decreases the value of CAPTCTRLX[CX1CNT] by 1 until 0 is reached. This register is not byte accessible.

28.3.35 Capture Value 1 Cycle Register (PWMA_SM*n*CVAL1CYC)

Address: E600h base + 23h offset + (48d × i), where i=0d to 3d

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	0												CVAL1CYC			
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PWMA_SM*n*CVAL1CYC field descriptions

Field	Description
15–4 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
CVAL1CYC	This read-only register stores the cycle number corresponding to the value captured in CVAL1. This register is incremented each time the counter is loaded with the INIT value at the end of a PWM modulo cycle.

28.3.36 Capture Value 2 Register (PWMA_SM*n*CVAL2)

Address: E600h base + 24h offset + (48d × i), where i=0d to 3d

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	CAPTVAL2															
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PWMA_SM*n*CVAL2 field descriptions

Field	Description
CAPTVAL2	This read-only register stores the value captured from the submodule counter. Exactly when this capture occurs is defined by CAPTCTRLA[EDGA0]. Each capture increases the value of CAPTCTRLA[CA0CNT] by 1 until the maximum value is reached. Each read of this register decreases the value of CAPTCTRLA[CA0CNT] by 1 until 0 is reached. This register is not byte accessible.

28.3.37 Capture Value 2 Cycle Register (PWMA_SM*n*CVAL2CYC)

Address: E600h base + 25h offset + (48d × i), where i=0d to 3d

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	0												CVAL2CYC			
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PWMA_SM*n*CVAL2CYC field descriptions

Field	Description
15–4 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
CVAL2CYC	This read-only register stores the cycle number corresponding to the value captured in CVAL2. This register is incremented each time the counter is loaded with the INIT value at the end of a PWM modulo cycle.

28.3.38 Capture Value 3 Register (PWMA_SM*n*CVAL3)

Address: E600h base + 26h offset + (48d × i), where i=0d to 3d

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	CAPTVAL3															
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PWMA_SM*n*CVAL3 field descriptions

Field	Description
CAPTVAL3	This read-only register stores the value captured from the submodule counter. Exactly when this capture occurs is defined by CAPTCTRLA[EDGA1]. Each capture increases the value of CAPTCTRLA[CA1CNT] by 1 until the maximum value is reached. Each read of this register decreases the value of CAPTCTRLA[CA1CNT] by 1 until 0 is reached. This register is not byte accessible.

28.3.39 Capture Value 3 Cycle Register (PWMA_SM*n*CVAL3CYC)

Address: E600h base + 27h offset + (48d × i), where i=0d to 3d

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	0												CVAL3CYC			
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PWMA_SM*n*CVAL3CYC field descriptions

Field	Description
15–4 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
CVAL3CYC	This read-only register stores the cycle number corresponding to the value captured in CVAL3. This register is incremented each time the counter is loaded with the INIT value at the end of a PWM modulo cycle.

28.3.40 Capture Value 4 Register (PWMA_SM*n*CVAL4)

Address: E600h base + 28h offset + (48d × i), where i=0d to 3d

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	CAPTVAL4															
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PWMA_SM*n*CVAL4 field descriptions

Field	Description
CAPTVAL4	This read-only register stores the value captured from the submodule counter. Exactly when this capture occurs is defined by CAPTCTRLB[EDGB0]. Each capture increases the value of CAPTCTRLB[CB0CNT] by 1 until the maximum value is reached. Each read of this register decreases the value of CAPTCTRLB[CB0CNT] by 1 until 0 is reached. This register is not byte accessible.

28.3.41 Capture Value 4 Cycle Register (PWMA_SM*n*CVAL4CYC)

Address: E600h base + 29h offset + (48d × i), where i=0d to 3d

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	0												CVAL4CYC			
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PWMA_SM*n*CVAL4CYC field descriptions

Field	Description
15–4 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
CVAL4CYC	This read-only register stores the cycle number corresponding to the value captured in CVAL4. This register is incremented each time the counter is loaded with the INIT value at the end of a PWM modulo cycle.

28.3.42 Capture Value 5 Register (PWMA_SM*n*CVAL5)

Address: E600h base + 2Ah offset + (48d × i), where i=0d to 3d

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	CAPTVAL5															
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PWMA_SM*n*CVAL5 field descriptions

Field	Description
CAPTVAL5	This read-only register stores the value captured from the submodule counter. Exactly when this capture occurs is defined by CAPTCTRLB[EDGB1]. Each capture increases the value of CAPTCTRLB[CB1CNT] by 1 until the maximum value is reached. Each read of this register decreases the value of CAPTCTRLB[CB1CNT] by 1 until 0 is reached. This register is not byte accessible.

28.3.43 Capture Value 5 Cycle Register (PWMA_SM*n*CVAL5CYC)

Address: E600h base + 2Bh offset + (48d × i), where i=0d to 3d

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	0												CVAL5CYC			
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PWMA_SM*n*CVAL5CYC field descriptions

Field	Description
15–4 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
CVAL5CYC	This read-only register stores the cycle number corresponding to the value captured in CVAL5. This register is incremented each time the counter is loaded with the INIT value at the end of a PWM modulo cycle.

28.3.44 Output Enable Register (PWMA_OUTEN)

Address: E600h base + C0h offset = E6C0h

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	0				PWMA_EN				PWMB_EN				PWMX_EN			
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PWMA_OUTEN field descriptions

Field	Description
15–12 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
11–8 PWMA_EN	PWM_A Output Enables The four bits of this field enable the PWM_A outputs of submodules 3-0, respectively. These bits should be set to 0 (output disabled) when a PWM_A pin is being used for input capture. 0 PWM_A output disabled. 1 PWM_A output enabled.
7–4 PWMB_EN	PWM_B Output Enables The four bits of this field enable the PWM_B outputs of submodules 3-0, respectively. These bits should be set to 0 (output disabled) when a PWM_B pin is being used for input capture. 0 PWM_B output disabled. 1 PWM_B output enabled.
PWMX_EN	PWM_X Output Enables The four bits of this field enable the PWM_X outputs of submodules 3-0, respectively. These bits should be set to 0 (output disabled) when a PWM_X pin is being used for input capture or deadtime correction. 0 PWM_X output disabled. 1 PWM_X output enabled.

28.3.45 Mask Register (PWMA_MASK)

MASK is double buffered and does not take effect until a FORCE_OUT event occurs within the appropriate submodule. Reading MASK reads the buffered values and not necessarily the values currently in effect.

Address: E600h base + C1h offset = E6C1h

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	0				MASKA				MASKB				MASKX			
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PWMA_MASK field descriptions

Field	Description
15–12 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
11–8 MASKA	PWM_A Masks The four bits of this field mask the PWM_A outputs of submodules 3-0, respectively, forcing the output to logic 0 prior to consideration of the output polarity. 0 PWM_A output normal. 1 PWM_A output masked.
7–4 MASKB	PWM_B Masks The four bits of this field mask the PWM_B outputs of submodules 3-0, respectively, forcing the output to logic 0 prior to consideration of the output polarity. 0 PWM_B output normal. 1 PWM_B output masked.
MASKX	PWM_X Masks The four bits of this field mask the PWM_X outputs of submodules 3-0, respectively, forcing the output to logic 0 prior to consideration of the output polarity. 0 PWM_X output normal. 1 PWM_X output masked.

28.3.46 Software Controlled Output Register (PWMA_SWCOUT)

These bits are double buffered and do not take effect until a FORCE_OUT event occurs within the appropriate submodule. Reading these bits reads the buffered value and not necessarily the value currently in effect.

Address: E600h base + C2h offset = E6C2h

Bit	15	14	13	12	11	10	9	8
Read	0							
Write								
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
Read Write	SM3OUT23	SM3OUT45	SM2OUT23	SM2OUT45	SM1OUT23	SM1OUT45	SM0OUT23	SM0OUT45
Reset	0	0	0	0	0	0	0	0

PWMA_SWCOUT field descriptions

Field	Description
15–8 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
7 SM3OUT23	Submodule 3 Software Controlled Output 23 This bit is only used when DTSRCSEL[SM3SEL23] is set to 0b10. It allows software control of which signal is supplied to the deadtime generator of that submodule. 0 A logic 0 is supplied to the deadtime generator of submodule 3 instead of PWM23. 1 A logic 1 is supplied to the deadtime generator of submodule 3 instead of PWM23.
6 SM3OUT45	Submodule 3 Software Controlled Output 45 This bit is only used when DTSRCSEL[SM3SEL45] is set to b10. It allows software control of which signal is supplied to the deadtime generator of that submodule. 0 A logic 0 is supplied to the deadtime generator of submodule 3 instead of PWM45. 1 A logic 1 is supplied to the deadtime generator of submodule 3 instead of PWM45.
5 SM2OUT23	Submodule 2 Software Controlled Output 23 This bit is only used when DTSRCSEL[SM2SEL23] is set to b10. It allows software control of which signal is supplied to the deadtime generator of that submodule. 0 A logic 0 is supplied to the deadtime generator of submodule 2 instead of PWM23. 1 A logic 1 is supplied to the deadtime generator of submodule 2 instead of PWM23.
4 SM2OUT45	Submodule 2 Software Controlled Output 45 This bit is only used when DTSRCSEL[SM2SEL45] is set to b10. It allows software control of which signal is supplied to the deadtime generator of that submodule. 0 A logic 0 is supplied to the deadtime generator of submodule 2 instead of PWM45. 1 A logic 1 is supplied to the deadtime generator of submodule 2 instead of PWM45.

Table continues on the next page...

PWMA_SWCOUT field descriptions (continued)

Field	Description
3 SM1OUT23	Submodule 1 Software Controlled Output 23 This bit is only used when DTSRCSEL[SM1SEL23] is set to b10. It allows software control of which signal is supplied to the deadtime generator of that submodule. 0 A logic 0 is supplied to the deadtime generator of submodule 1 instead of PWM23. 1 A logic 1 is supplied to the deadtime generator of submodule 1 instead of PWM23.
2 SM1OUT45	Submodule 1 Software Controlled Output 45 This bit is only used when DTSRCSEL[SM1SEL45] is set to b10. It allows software control of which signal is supplied to the deadtime generator of that submodule. 0 A logic 0 is supplied to the deadtime generator of submodule 1 instead of PWM45. 1 A logic 1 is supplied to the deadtime generator of submodule 1 instead of PWM45.
1 SM0OUT23	Submodule 0 Software Controlled Output 23 This bit is only used when DTSRCSEL[SM0SEL23] is set to b10. It allows software control of which signal is supplied to the deadtime generator of that submodule. 0 A logic 0 is supplied to the deadtime generator of submodule 0 instead of PWM23. 1 A logic 1 is supplied to the deadtime generator of submodule 0 instead of PWM23.
0 SM0OUT45	Submodule 0 Software Controlled Output 45 This bit is only used when DTSRCSEL[SM0SEL45] is set to b10. It allows software control of which signal is supplied to the deadtime generator of that submodule. 0 A logic 0 is supplied to the deadtime generator of submodule 0 instead of PWM45. 1 A logic 1 is supplied to the deadtime generator of submodule 0 instead of PWM45.

28.3.47 PWM Source Select Register (PWMA_DTSRCSEL)

The PWM source select bits are double buffered and do not take effect until a FORCE_OUT event occurs within the appropriate submodule. Reading these bits reads the buffered value and not necessarily the value currently in effect.

Address: E600h base + C3h offset = E6C3h

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read Write	SM3SEL23		SM3SEL45		SM2SEL23		SM2SEL45		SM1SEL23		SM1SEL45		SM0SEL23		SM0SEL45	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PWMA_DTSRCSEL field descriptions

Field	Description
15–14 SM3SEL23	Submodule 3 PWM23 Control Select This field selects possible over-rides to the generated SM3PWM23 signal that will be passed to the deadtime logic upon the occurrence of a FORCE_OUT event in that submodule.

Table continues on the next page...

PWMA_DTSRCSEL field descriptions (continued)

Field	Description
	00 Generated SM3PWM23 signal is used by the deadtime logic. 01 Inverted generated SM3PWM23 signal is used by the deadtime logic. 10 SWCOUT[SM3OUT23] is used by the deadtime logic. 11 PWM3_EXTA signal is used by the deadtime logic.
13–12 SM3SEL45	Submodule 3 PWM45 Control Select This field selects possible over-rides to the generated SM3PWM45 signal that will be passed to the deadtime logic upon the occurrence of a FORCE_OUT event in that submodule. 00 Generated SM3PWM45 signal is used by the deadtime logic. 01 Inverted generated SM3PWM45 signal is used by the deadtime logic. 10 SWCOUT[SM3OUT45] is used by the deadtime logic. 11 PWM3_EXTB signal is used by the deadtime logic.
11–10 SM2SEL23	Submodule 2 PWM23 Control Select This field selects possible over-rides to the generated SM2PWM23 signal that will be passed to the deadtime logic upon the occurrence of a FORCE_OUT event in that submodule. 00 Generated SM2PWM23 signal is used by the deadtime logic. 01 Inverted generated SM2PWM23 signal is used by the deadtime logic. 10 SWCOUT[SM2OUT23] is used by the deadtime logic. 11 PWM2_EXTA signal is used by the deadtime logic.
9–8 SM2SEL45	Submodule 2 PWM45 Control Select This field selects possible over-rides to the generated SM2PWM45 signal that will be passed to the deadtime logic upon the occurrence of a FORCE_OUT event in that sudeadtime logic upon the occurrence of a FORCE_OUT event in that submodule. 00 Generated SM2PWM45 signal is used by the deadtime logic. 01 Inverted generated SM2PWM45 signal is used by the deadtime logic. 10 SWCOUT[SM2OUT45] is used by the deadtime logic. 11 PWM2_EXTB signal is used by the deadtime logic.
7–6 SM1SEL23	Submodule 1 PWM23 Control Select This field selects possible over-rides to the generated SM1PWM23 signal that will be passed to the deadtime logic upon the occurrence of a FORCE_OUT event in that submodule. 00 Generated SM1PWM23 signal is used by the deadtime logic. 01 Inverted generated SM1PWM23 signal is used by the deadtime logic. 10 SWCOUT[SM1OUT23] is used by the deadtime logic. 11 PWM1_EXTA signal is used by the deadtime logic.
5–4 SM1SEL45	Submodule 1 PWM45 Control Select This field selects possible over-rides to the generated SM1PWM45 signal that will be passed to the deadtime logic upon the occurrence of a FORCE_OUT event in that submodule. 00 Generated SM1PWM45 signal is used by the deadtime logic. 01 Inverted generated SM1PWM45 signal is used by the deadtime logic. 10 SWCOUT[SM1OUT45] is used by the deadtime logic. 11 PWM1_EXTB signal is used by the deadtime logic.

Table continues on the next page...

PWMA_DTSRCSEL field descriptions (continued)

Field	Description
3–2 SM0SEL23	Submodule 0 PWM23 Control Select This field selects possible over-rides to the generated SM0PWM23 signal that will be passed to the deadtime logic upon the occurrence of a FORCE_OUT event in that submodule. 00 Generated SM0PWM23 signal is used by the deadtime logic. 01 Inverted generated SM0PWM23 signal is used by the deadtime logic. 10 SWCOUT[SM0OUT23] is used by the deadtime logic. 11 PWM0_EXTA signal is used by the deadtime logic.
SM0SEL45	Submodule 0 PWM45 Control Select This field selects possible over-rides to the generated SM0PWM45 signal that will be passed to the deadtime logic upon the occurrence of a FORCE_OUT event in that submodule. 00 Generated SM0PWM45 signal is used by the deadtime logic. 01 Inverted generated SM0PWM45 signal is used by the deadtime logic. 10 SWCOUT[SM0OUT45] is used by the deadtime logic. 11 PWM0_EXTB signal is used by the deadtime logic.

28.3.48 Master Control Register (PWMA_MCTRL)

In every 4-bit field in this register, each bit acts on a separate submodule. Accordingly, the description of every bitfield refers to the effect of an individual bit.

Address: E600h base + C4h offset = E6C4h

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	IPOL				RUN				0				LDOK			
Write									CLDOK							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PWMA_MCTRL field descriptions

Field	Description
15–12 IPOL	Current Polarity The four buffered read/write bits of this field correspond to submodules 3-0, respectively. Each bit selects between PWM23 and PWM45 as the source for the generation of the complementary PWM pair output for the corresponding submodule. MCTRL[IPOL] is ignored in independent mode. MCTRL[IPOL] does not take effect until a FORCE_OUT event takes place in the appropriate submodule. Reading MCTRL[IPOL] reads the buffered value and not necessarily the value currently in effect. 0 PWM23 is used to generate complementary PWM pair in the corresponding submodule. 1 PWM45 is used to generate complementary PWM pair in the corresponding submodule.

Table continues on the next page...

PWMA_MCTRL field descriptions (continued)

Field	Description
11–8 RUN	Run The four read/write bits of this field enable the clocks to the PWM generator of submodules 3-0, respectively. The corresponding MCTRL[RUN] bit must be set for each submodule that is using its input capture functions or is using the local reload as its reload source. When this bit equals zero, the submodule counter is reset. A reset clears this field. 0 PWM generator is disabled in the corresponding submodule. 1 PWM generator is enabled in the corresponding submodule.
7–4 CLDOK	Clear Load Okay The 4 bits of CLDOK field correspond to submodules 3-0, respectively. Each write-only bit is used to clear the corresponding bit of MCTRL[LDOK]. Write a 1 to CLDOK to clear the corresponding MCTRL[LDOK] bit. If a reload occurs within a submodule with the corresponding MCTRL[LDOK] bit set at the same time that MCTRL[CLDOK] is written, then the reload in that submodule will not be performed and MCTRL[LDOK] will be cleared. CLDOK bit is self-clearing and always reads as a 0.
LDOK	Load Okay The 4 bits of LDOK field correspond to submodules 3-0, respectively. Each read/set bit loads CTRL[PRSC] and the INIT, FRACVALx, and VALx registers of the corresponding submodule into a set of buffers. The buffered prescaler divisor, submodule counter modulus value, and PWM pulse width take effect at the next PWM reload if CTRL[LDMOD] is clear or immediately if CTRL[LDMOD] is set. Set the corresponding MCTRL[LDOK] bit by reading it when it is logic zero and then writing a logic one to it. The VALx, FRACVALx,INIT, and CTRL[PRSC] registers of the corresponding submodule cannot be written while the the corresponding MCTRL[LDOK] bit is set. In Master Reload Mode (CTRL2[RELOAD_SEL]=1), it is only necessary to set the LDOK bit corresponding to submodule0; however, it is recommended to also set the LDOK bit of the slave submodules, to prevent unwanted writes to the registers in the slave submodules. The MCTRL[LDOK] bit is automatically cleared after the new values are loaded, or it can be manually cleared before a reload by writing a logic 1 to the appropriate MCTRL[CLDOK] bit. LDOK bits cannot be written with a zero. MCTRL[LDOK] can be set in DMA mode when the DMA indicates that it has completed the update of all CTRL[PRSC], INIT,FRACVALx, and VALx registers in the corresponding submodule. Reset clears LDOK field. 0 Do not load new values. 1 Load prescaler, modulus, and PWM values of the corresponding submodule.

28.3.49 Master Control 2 Register (PWMA_MCTRL2)

Address: E600h base + C5h offset = E6C5h

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	0														MONPLL	
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PWMA_MCTRL2 field descriptions

Field	Description
15–2 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
MONPLL	Monitor PLL State These bits are used to control disabling of the fractional delay block when the chip PLL is unlocked and/or missing its input reference. The fractional delay block requires a continuous 200 MHz clock from the PLL. If this clock turns off when the fractional delay block is being used, then the output of the fractional delay block can be stuck high or low even if the PLL restarts. When this control bit is set, PLL problems cause the fractional delay block to be disabled until the PLL returns to a locked state. Once the PLL is receiving a proper reference and is locked, the fractional delay block requires a 25 µs startup time just as if the FRCTRL[FRAC*_EN] bits had been turned off and turned on again. If PLL monitoring is disabled, then software should manually clear and then set the FRCTRL[FRAC*_EN] bits when the PLL loses its reference or loses lock. This will cause the fractional delay block to be disabled and restarted. If the fractional delay block is not being used, then the value of these bits do not matter. 00 Not locked. Do not monitor PLL operation. Resetting of the fractional delay block in case of PLL losing lock will be controlled by software. 01 Not locked. Monitor PLL operation to automatically disable the fractional delay block when the PLL encounters problems. 10 Locked. Do not monitor PLL operation. Resetting of the fractional delay block in case of PLL losing lock will be controlled by software. These bits are write protected until the next reset. 11 Locked. Monitor PLL operation to automatically disable the fractional delay block when the PLL encounters problems. These bits are write protected until the next reset.

28.3.50 Fault Control Register (PWMA_FCTRL*n*)

For every 4-bit field in this register, the bits act on the fault inputs in order. For example, FLVL bits 15-12 act on faults 3-0, respectively.

Address: E600h base + C6h offset + (4d × i), where i=0d to 1d

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read Write	FLVL				FAUTO				FSAFE				FIE			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PWMA_FCTRL*n* field descriptions

Field	Description
15–12 FLVL	Fault Level The four read/write bits of this field select the active logic level of the individual fault inputs 3-0, respectively. A reset clears this field. 0 A logic 0 on the fault input indicates a fault condition. 1 A logic 1 on the fault input indicates a fault condition.

Table continues on the next page...

PWMA_FCTRL*n* field descriptions (continued)

Field	Description
11–8 FAUTO	Automatic Fault Clearing The four read/write bits of this field select automatic or manual clearing of faults 3-0, respectively. A reset clears this field. 0 Manual fault clearing. PWM outputs disabled by this fault are not enabled until FSTS[FFLAGx] is clear at the start of a half cycle or full cycle depending the state of FSTS[FFULL]. This is further controlled by FCTRL[FSAFE]. 1 Automatic fault clearing. PWM outputs disabled by this fault are enabled when FSTS[FFPINx] is clear at the start of a half cycle or full cycle depending on the state of FSTS[FFULL] without regard to the state of FSTS[FFLAGx].
7–4 FSAFE	Fault Safety Mode These read/write bits select the safety mode during manual fault clearing. A reset clears this field. FSTS[FFPINx] may indicate a fault condition still exists even though the actual fault signal at the FAULTx pin is clear due to the fault filter latency. 0 Normal mode. PWM outputs disabled by this fault are not enabled until FSTS[FFLAGx] is clear at the start of a half cycle or full cycle depending on the state of FSTS[FFULL] without regard to the state of FSTS[FFPINx]. The PWM outputs disabled by this fault input will not be re-enabled until the actual FAULTx input signal de-asserts since the fault input will combinationally disable the PWM outputs (as programmed in DISMAPn). 1 Safe mode. PWM outputs disabled by this fault are not enabled until FSTS[FFLAGx] is clear and FSTS[FFPINx] is clear at the start of a half cycle or full cycle depending on the state of FSTS[FFULL].
FIE	Fault Interrupt Enables This read/write field enables CPU interrupt requests generated by the FAULTx pins. A reset clears this field. **NOTE:** The fault protection circuit is independent of the FIEx bit and is always active. If a fault is detected, the PWM outputs are disabled according to the disable mapping register. 0 FAULTx CPU interrupt requests disabled. 1 FAULTx CPU interrupt requests enabled.

28.3.51 Fault Status Register (PWMA_FSTS*n*)

Address: E600h base + C7h offset + (4d × i), where i=0d to 1d

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	FHALF				FFPIN				FFULL				FFLAG			
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PWMA_FSTS*n* field descriptions

Field	Description
15–12 FHALF	Half Cycle Fault Recovery These read/write bits are used to control the timing for re-enabling the PWM outputs after a fault condition. These bits apply to both automatic and manual clearing of a fault condition.

Table continues on the next page...

PWMA_FSTS*n* field descriptions (continued)

Field	Description
	NOTE: Both FHALF and FFULL can be set so that the fault recovery occurs at the start of a full cycle and at the start of a half cycle (as defined by VAL0). If neither FHALF nor FFULL is set, then no fault recovery is possible. 0 PWM outputs are not re-enabled at the start of a half cycle. 1 PWM outputs are re-enabled at the start of a half cycle (as defined by VAL0).
11–8 FFPIN	Filtered Fault Pins These read-only bits reflect the current state of the filtered FAULTx pins converted to high polarity. A logic 1 indicates a fault condition exists on the filtered FAULTx pin. A reset has no effect on this field.
7–4 FFULL	Full Cycle These read/write bits are used to control the timing for re-enabling the PWM outputs after a fault condition. These bits apply to both automatic and manual clearing of a fault condition. **NOTE:** Both FHALF and FFULL can be set so that the fault recovery occurs at the start of a full cycle and at the start of a half cycle (as defined by VAL0). If neither FHALF nor FFULL is set, then no fault recovery is possible. 0 PWM outputs are not re-enabled at the start of a full cycle 1 PWM outputs are re-enabled at the start of a full cycle
FFLAG	Fault Flags These read-only flag is set within two CPU cycles after a transition to active on the FAULTx pin. Clear this bit by writing a logic one to it. A reset clears this field. While the reset value is 0, these bits may be set to 1 by the time they can be read depending on the state of the fault input signals. 0 No fault on the FAULTx pin. 1 Fault on the FAULTx pin.

28.3.52 Fault Filter Register (PWMA_FFILT*n*)

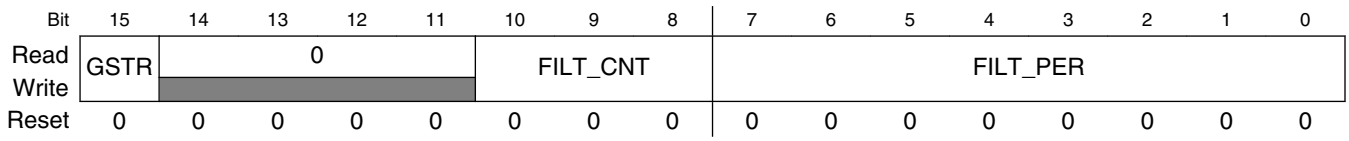
The settings in this register are shared among each of the fault input filters within the fault channel.

Input filter considerations include:

- The FILT_PER value should be set such that the sampling period is larger than the period of the expected noise. This way a noise spike will only corrupt one sample. The FILT_CNT value should be chosen to reduce the probability of noisy samples causing an incorrect transition to be recognized. The probability of an incorrect transition is defined as the probability of an incorrect sample raised to the FILT_CNT+3 power.
- The values of FILT_PER and FILT_CNT must also be traded off against the desire for minimal latency in recognizing input transitions. Turning on the input filter (setting FILT_PER to a non-zero value) introduces a latency of ((FILT_CNT+4) x FILT_PER x IPBus clock period). Note that even when the filter is enabled, there is a combinational path to disable the PWM outputs. This is to ensure rapid response to

fault conditions and also to ensure fault response if the PWM module loses its clock. The latency induced by the filter will be seen in the time to set FSTS[FFLAG] and FSTS[FFPIN].

Address: E600h base + C8h offset + (4d × i), where i=0d to 1d

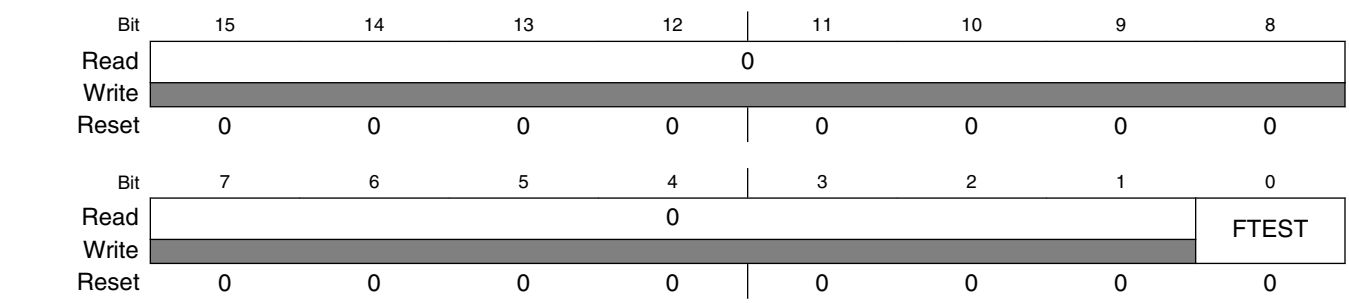


PWMA_FFILT*n* field descriptions

Field	Description
15 GSTR	Fault Glitch Stretch Enable This bit is used to enable the fault glitch stretching logic. This logic ensures that narrow fault glitches are stretched to be at least 2 IPBus clock cycles wide. In some cases a narrow fault input can cause problems due to the short PWM output shutdown/re-activation time. The stretching logic ensures that a glitch on the fault input, when the fault filter is disabled, will be registered in the fault flags. 0 Fault input glitch stretching is disabled. 1 Input fault signals will be stretched to at least 2 IPBus clock cycles.
14–11 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
10–8 FILT_CNT	Fault Filter Count These bits represent the number of consecutive samples that must agree prior to the input filter accepting an input transition. The number of samples is the decimal value of this field plus three: the bitfield value of 0-7 represents 3-10 samples, respectively. The value of FILT_CNT affects the input latency.
FILT_PER	Fault Filter Period This 8-bit field applies universally to all fault inputs. These bits represent the sampling period (in IPBus clock cycles) of the fault pin input filter. Each input is sampled multiple times at the rate specified by this field. If FILT_PER is 0x00 (default), then the input filter is bypassed. The value of FILT_PER affects the input latency. **NOTE:** When changing values for FILT_PER from one non-zero value to another non-zero value, first write a value of zero to clear the filter.

28.3.53 Fault Test Register (PWMA_FTST*n*)

Address: E600h base + C9h offset + (4d × i), where i=0d to 1d



PWMA_FTST*n* field descriptions

Field	Description
15–1 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
0 FTEST	Fault Test This read/write bit is used to simulate a fault condition. Setting this bit causes a simulated fault to be sent into all of the fault filters. The condition propagates to the fault flags and possibly the PWM outputs depending on the DISMAPn settings. Clearing this bit removes the simulated fault condition. 0 No fault 1 Cause a simulated fault

28.4 Functional Description

28.4.1 PWM Capabilities

This section describes some capabilities of the PWM module.

28.4.1.1 Center Aligned PWMs

Each submodule has its own timer that is capable of generating PWM signals on two output pins. The edges of each of these signals are controlled independently as shown in Figure 28-3.

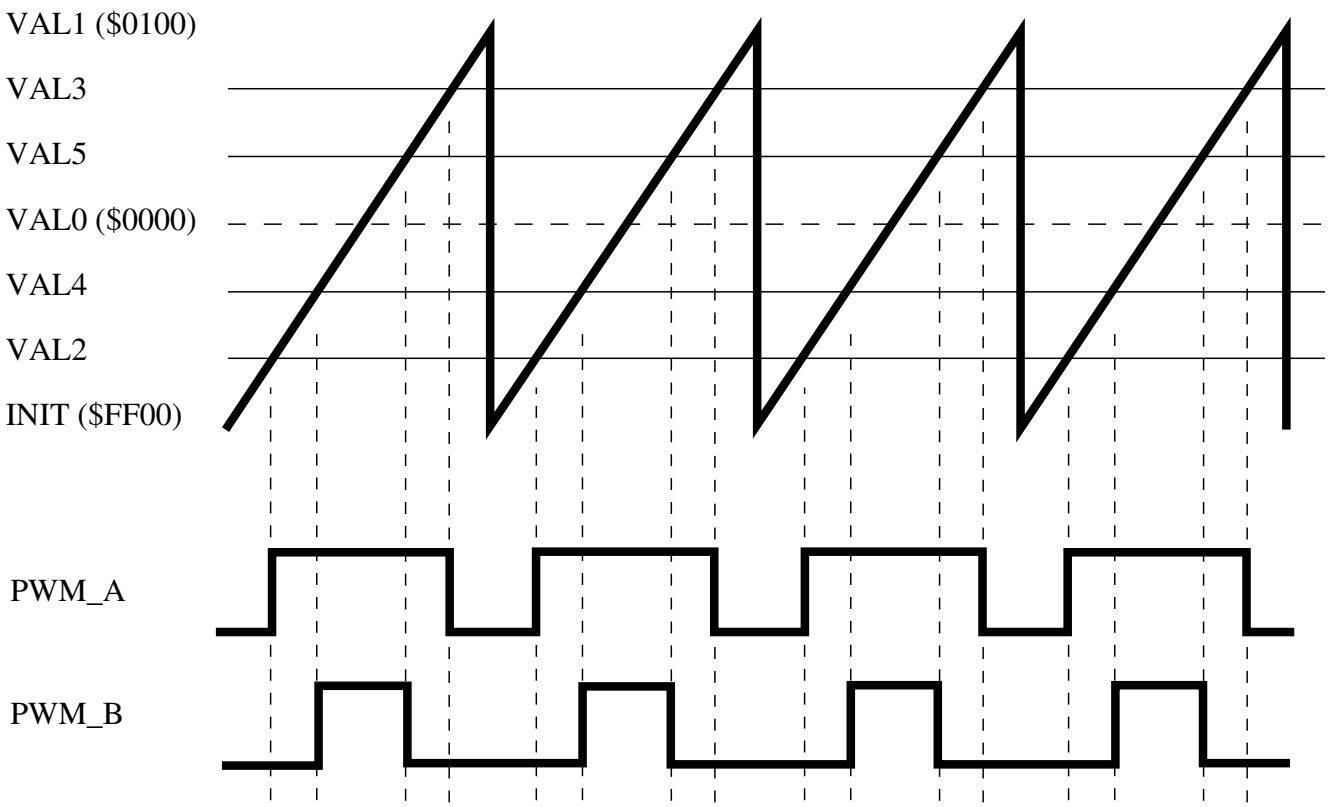


Figure 28-3. Center Aligned Example

The submodule timers only count in the up direction and then reset to the INIT value. Instead of having a single value that determines pulse width, there are two values that must be specified: the turn on edge and the turn off edge. This double action edge generation not only gives the user control over the pulse width, but over the relative alignment of the signal as well. As a result, there is no need to support separate PWM alignment modes since the PWM alignment mode is inherently a function of the turn on and turn off edge values.

Figure 28-3 also illustrates an additional enhancement to the PWM generation process. When the counter resets, it is reloaded with a user specified value, which may or may not be zero. If the value chosen happens to be the 2's complement of the modulus value, then the PWM generator operates in "signed" mode. This means that if each PWM's turn on and turn off edge values are also the same number but only different in their sign, the "on" portion of the output signal will be centered around a count value of zero. Therefore, only one PWM value needs to be calculated in software and then this value and its negative are provided to the submodule as the turn off and turn on edges respectively. This technique will result in a pulse width that always consists of an odd number of timer counts. If all PWM signal edge calculations follow this same convention, then the signals will be center aligned with respect to each other, which is the goal. Of course, center

alignment between the signals is not restricted to symmetry around the zero count value, as any other number would also work. However, centering on zero provides the greatest range in signed mode and also simplifies the calculations.

28.4.1.2 Edge Aligned PWMs

When the turn on edge for each pulse is specified to be the INIT value, then edge aligned operation results, as the following figure shows. Therefore, only the turn off edge value needs to be periodically updated to change the pulse width.

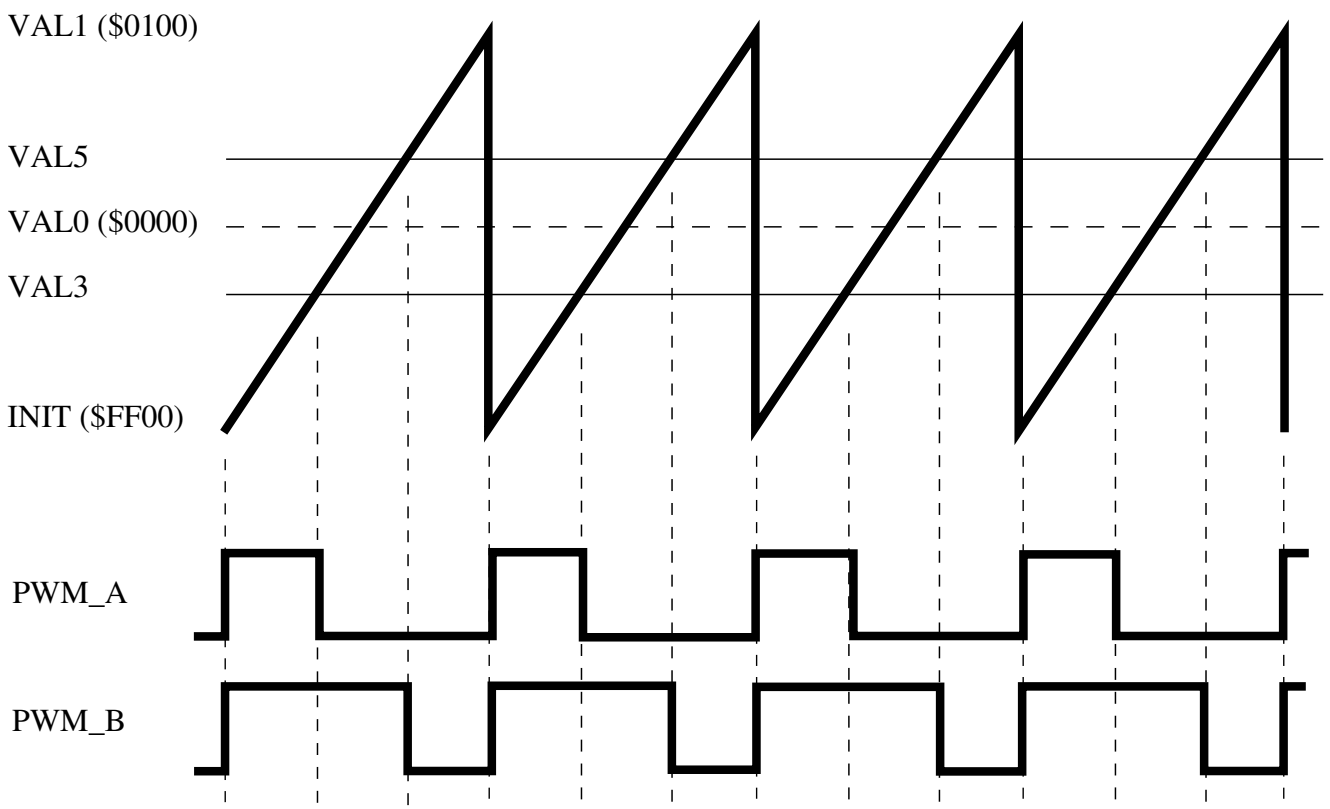


Figure 28-4. Edge Aligned Example (INIT=VAL2=VAL4)

With edge aligned PWMs, another example of the benefits of signed mode can be seen. A common way to drive an H-bridge is to use a technique called "bipolar" PWMs where a 50% duty cycle results in zero volts on the load. Duty cycles less than 50% result in negative load voltages and duty cycles greater than 50% generate positive load voltages. If the module is set to signed mode operation (the INIT and VAL1 values are the same number with opposite signs), then there is a direct proportionality between the PWM turn off edge value and the inverter voltage, INCLUDING the sign. So once again, signed mode of operation simplifies the software interface to the PWM module since no offset calculations are required to translate the output variable control algorithm to the voltage on an H-Bridge load.

28.4.1.3 Phase Shifted PWMs

In the previous sections, the benefits of signed mode of operation were discussed in the context of simplifying the required software calculations by eliminating the requirement to bias up signed variables before applying them to the module. However, if numerical biases are applied to the turn on and turn off edges of different PWM signal, the signals will be phase shifted with respect to each other, as the following figure shows. This results in certain advantages when applied to a power stage. For example, when operating a multi-phase inverter at a low modulation index, all of the PWM switching edges from the different phases occur at nearly the same time. This can be troublesome from a noise standpoint, especially if ADC readings of the inverter must be scheduled near those times. Phase shifting the PWM signals can open up timing windows between the switching edges to allow a signal to be sampled by the ADC. However, phase shifting does NOT affect the duty cycle so average load voltage is not affected.

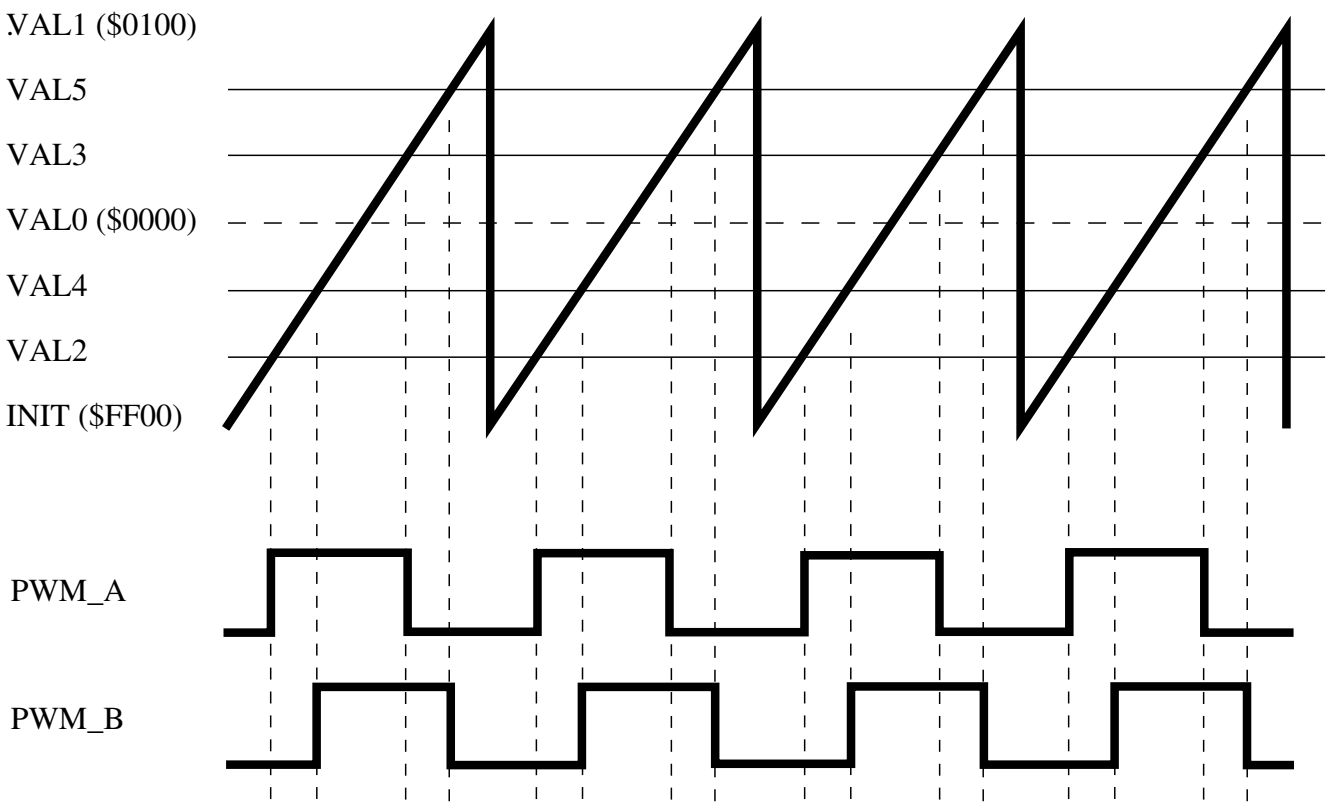


Figure 28-5. Phase Shifted Outputs Example

An additional benefit of phase shifted PWMs appears in Figure 28-6. In this case, an H-Bridge circuit is driven by 4 PWM signals to control the voltage waveform on the primary of a transformer. Both left and right side PWMs are configured to always generate a square wave with 50% duty cycle. This works out nicely for the H-Bridge since no narrow pulse widths are generated reducing the high frequency switching

requirements of the transistors. Notice that the square wave on the right side of the H-Bridge is phase shifted compared to the left side of the H-Bridge. As a result, the transformer primary sees the bottom waveform across its terminals. The RMS value of this waveform is directly controlled by the amount of phase shift of the square waves. Regardless of the phase shift, no DC component appears in the load voltage as long as the duty cycle of each square wave remains at 50% making this technique ideally suited for transformer loads. As a result, this topology is frequently used in industrial welders to adjust the amount of energy delivered to the weld arc.

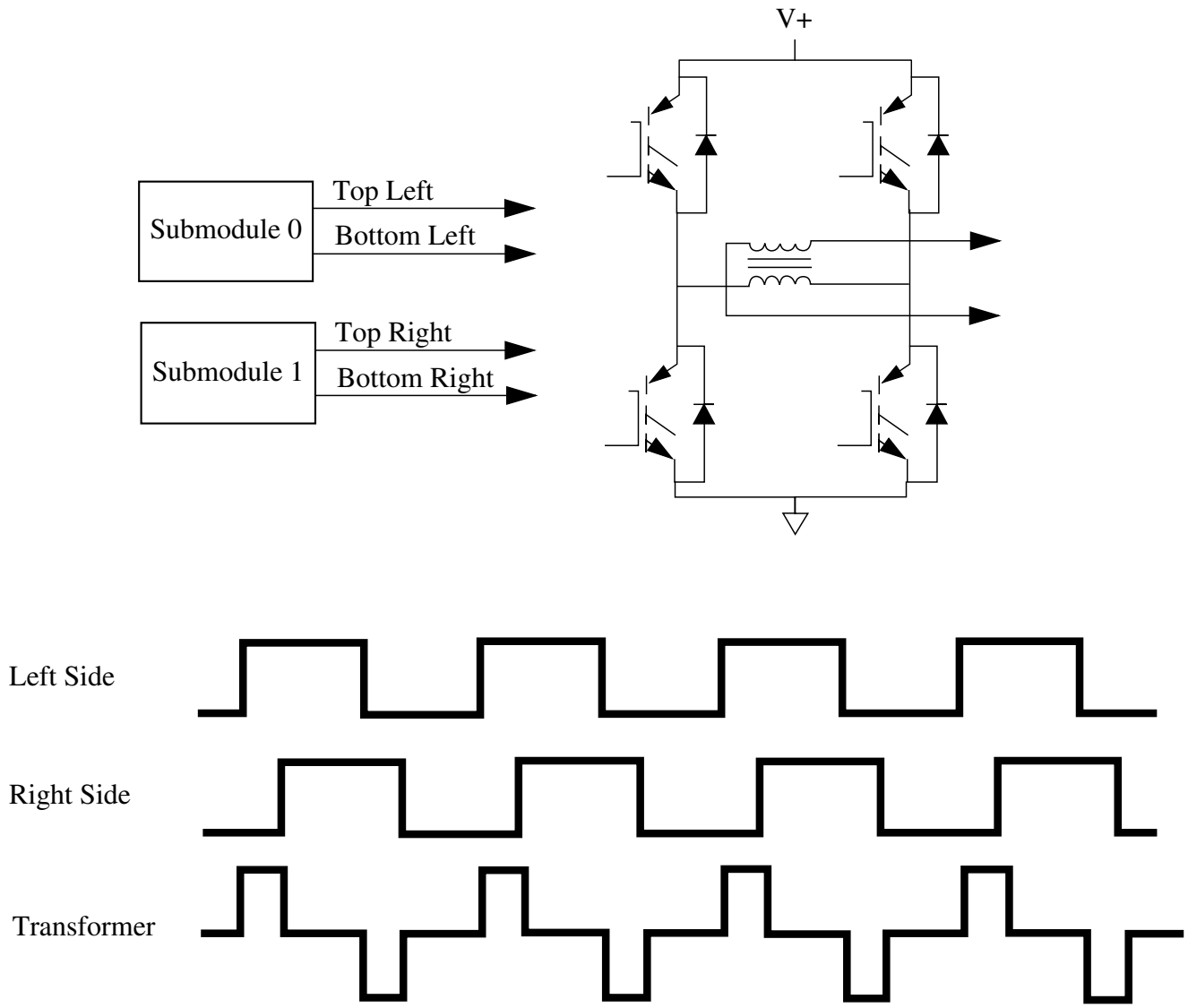


Figure 28-6. Phase Shifted PWMs Applied to a Transformer Primary

28.4.1.4 Double Switching PWMs

Double switching PWM output is supported to aid in single shunt current measurement and three phase reconstruction. This method support two independent rising edges and two independent falling edges per PWM cycle. The VAL2 and VAL3 registers are used to generate the even channel (labelled as PWM_A in the figure) while VAL4 and VAL5 are used to generate the odd channel. The two channels are combined using XOR logic (force out logic) as the following figure shows. The DBLPWM signal can be run through the deadtime insertion logic.

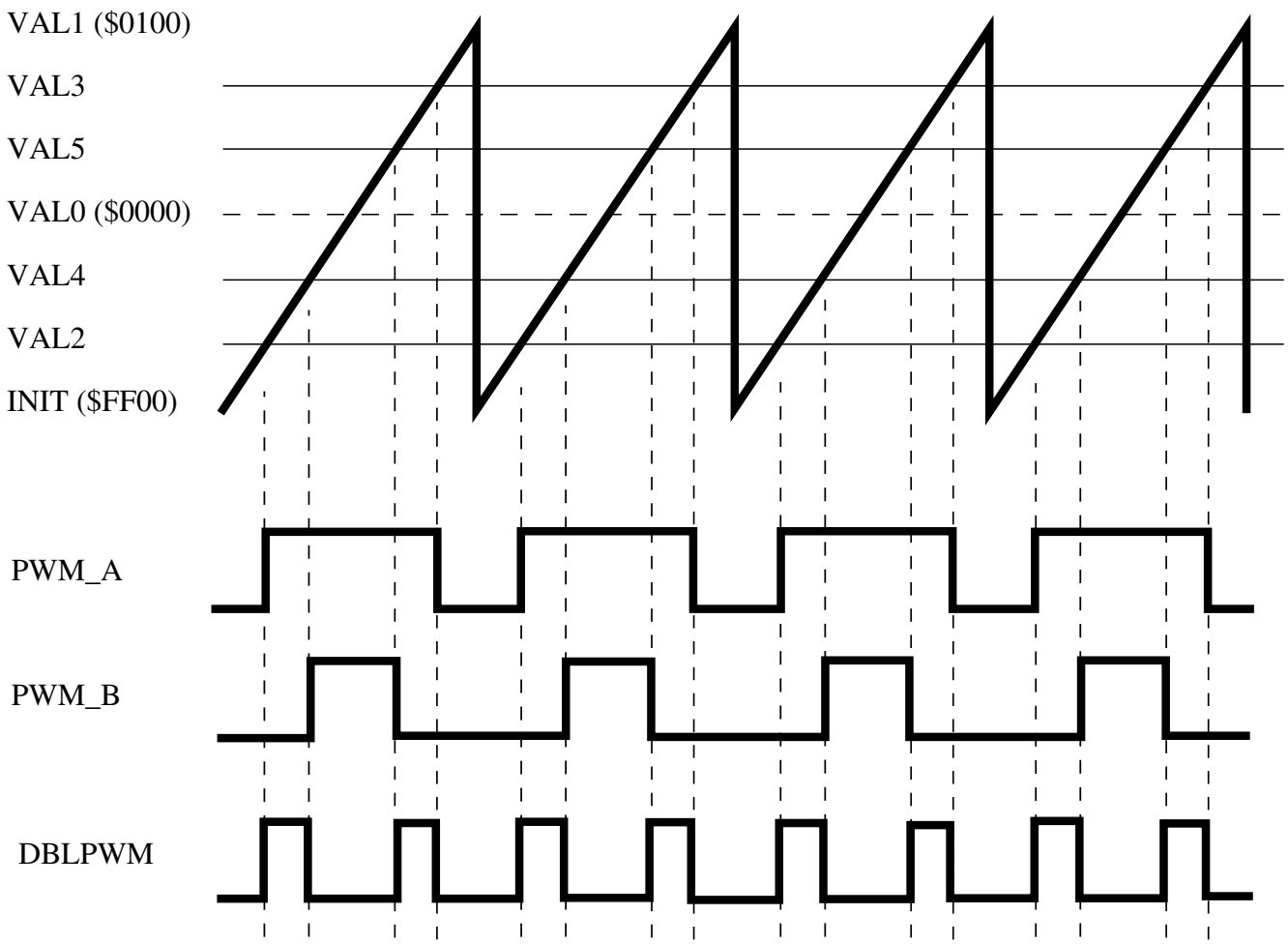


Figure 28-7. Double Switching Output Example

28.4.1.5 ADC Triggering

In cases where the timing of the ADC triggering is critical, it must be scheduled as a hardware event instead of software activated. With this PWM module, multiple ADC triggers can be generated in hardware per PWM cycle without the requirement of another timer module. Figure 28-8 shows how this is accomplished. When specifying

complementary mode of operation, only two edge comparators are required to generate the output PWM signals for a given submodule. This means that the other comparators are free to perform other functions. In this example, the software does not need to quickly respond after the first conversion to set up other conversions that must occur in the same PWM cycle.

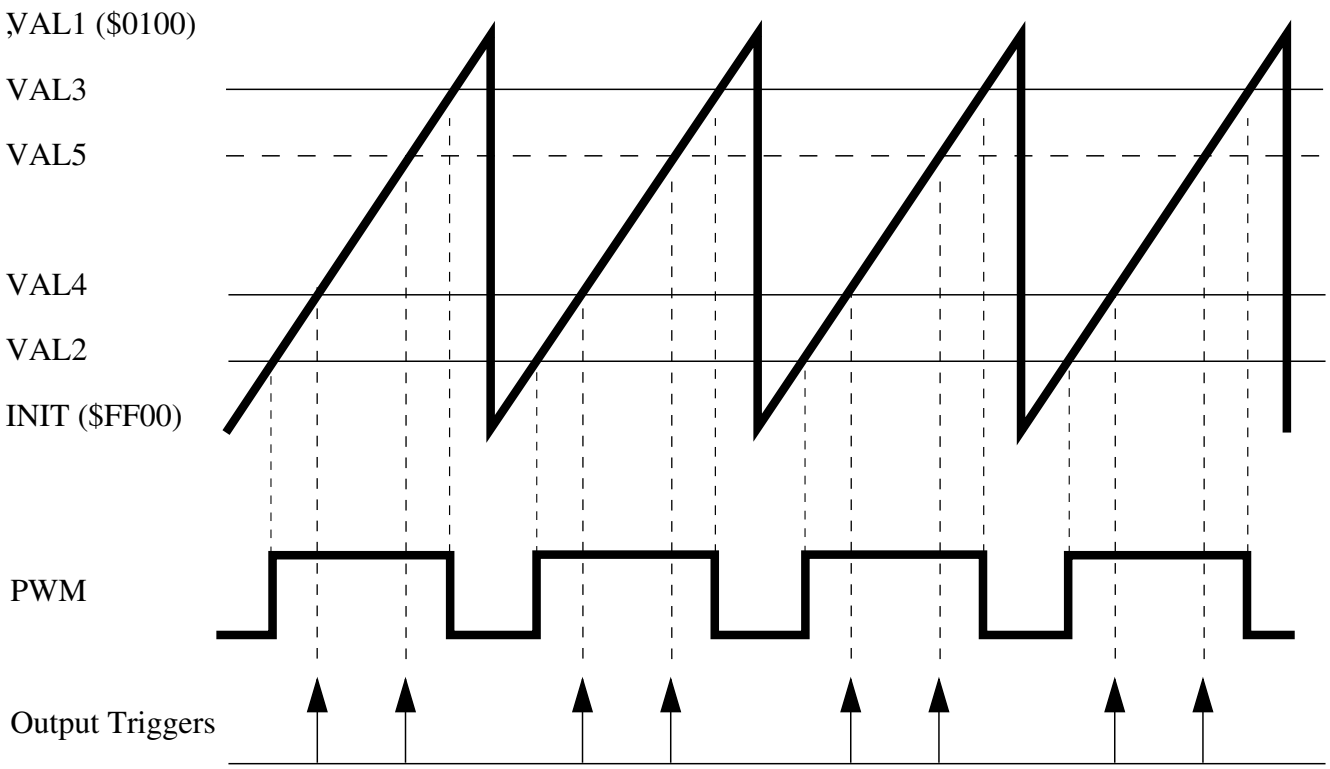


Figure 28-8. Multiple Output Trigger Generation in Hardware

Because each submodule has its own timer, it is possible for each submodule to run at a different frequency. One of the options possible with this PWM module is to have one or more submodules running at a lower frequency, but still synchronized to the timer in submodule0. Figure 28-9 shows how this feature can be used to schedule ADC triggers over multiple PWM cycles. A suggested use for this configuration would be to use the lower-frequency submodule to control the sampling frequency of the software control algorithm where multiple ADC triggers can now be scheduled over the entire sampling period. In Figure 28-9, *all* submodule comparators are shown being used for ADC trigger generation.

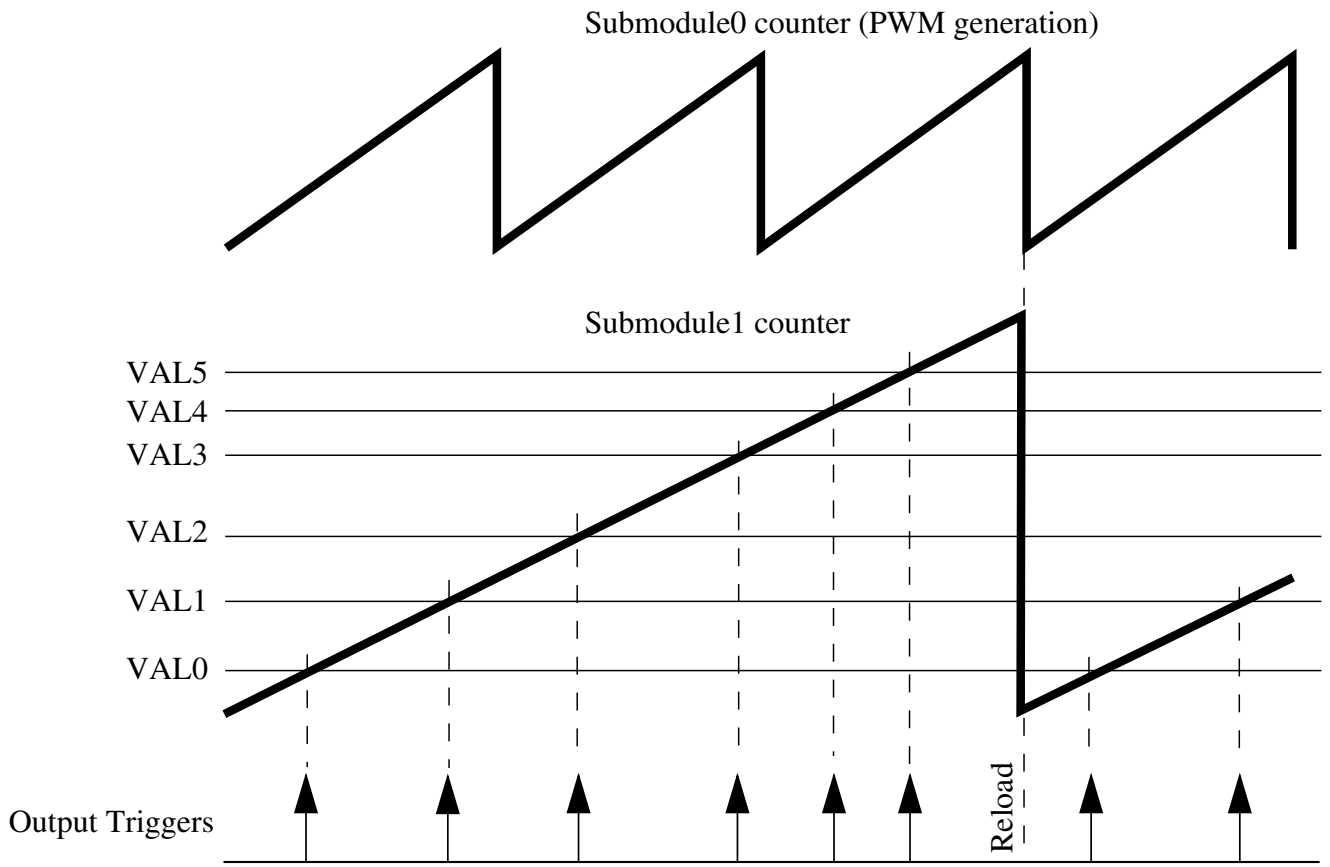


Figure 28-9. Multiple Output Triggers Over Several PWM Cycles

28.4.1.6 Enhanced Capture Capabilities (E-Capture)

When a PWM pin is not being used for PWM generation, it can be used to perform input captures. Recall that for PWM generation BOTH edges of the PWM signal are specified via separate compare register values. When programmed for input capture, both of these registers work on the same pin to capture multiple edges, toggling from one to the other in either a free running or one-shot fashion. By simply programming the desired edge of each capture circuit, period and pulse width of an input signal can easily be measured without the requirement to re-arm the circuit. In addition, each edge of the input signal can clock an 8 bit counter where the counter output is compared to a user specified value (EDGCMP). When the counter output equals EDGCMP, the value of the submodule timer is captured and the counter is automatically reset. This feature allows the module to count a specified number of edge events and then perform a capture and interrupt. The following figure illustrates some of the functionality of the E-Capture circuit.

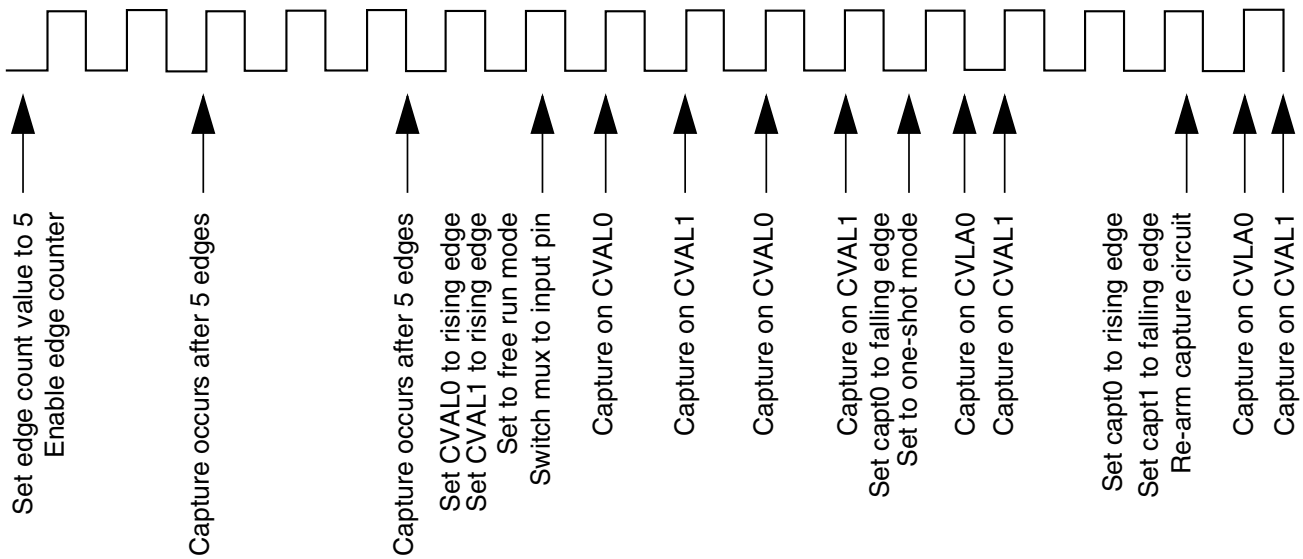


Figure 28-10. Capture Capabilities of the E-Capture Circuit

When a submodule is being used for PWM generation, its timer counts up to the modulus value used to specify the PWM frequency and then is re-initialized. Therefore, using this timer for input captures on one of the other pins (for example, PWM_X) has limited utility since it does not count through all of the numbers and the timer reset represents a discontinuity in the 16 bit number range. However, when measuring a signal that is synchronous to the PWM frequency, the timer modulus range is perfectly suited for the application. Consider the following figure as an example. In this application the output of a PWM power stage is connected to the PWM_X pin that is configured for free running input captures. Specifically, the CVAL0 capture circuitry is programmed for rising edges and the CVAL1 capture circuitry is set for falling edges. This will result in new load pulse width data being acquired every PWM cycle. To calculate the pulse width, simply subtract the CVAL0 register value from the CVAL1 register value. This measurement is extremely beneficial when performing dead-time distortion correction on a half bridge circuit driving an inductive load. Also, these values can be directly compared to the VALx registers responsible for generating the PWM outputs to obtain a measurement of system propagation delays. For details, refer to the separate discussion of deadtime distortion correction.

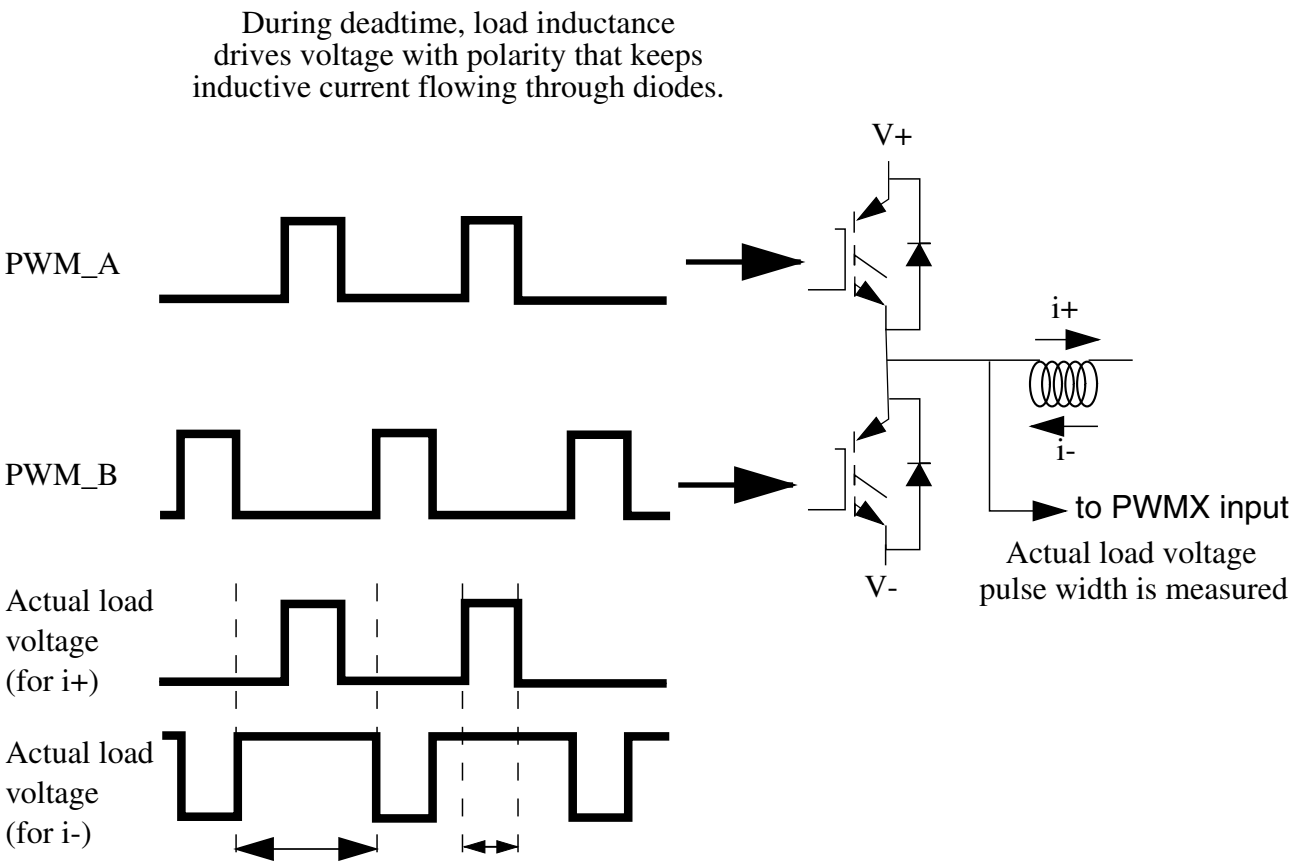


Figure 28-11. Output Pulse Width Measurement Possible with the E-Capture Circuit

28.4.1.7 Synchronous Switching of Multiple Outputs

Before the PWM signals are routed to the output pins, they are processed by a hardware block that permits all submodule outputs to be switched synchronously. Not only do all the changes occur synchronously on all submodule outputs, but they occur IMMEDIATELY after the trigger event occurs eliminating any interrupt latency.

The synchronous output switching is accomplished via a signal called FORCE_OUT. This signal originates from the local FORCE bit within the submodule, from submodule0, or from external to the PWM module and, in most cases, is supplied from an external timer channel configured for output compare. In a typical application, software sets up the desired states of the output pins in preparation for the next FORCE_OUT event. This selection lays dormant until the FORCE_OUT signal transitions and then all outputs are switched simultaneously. The signal switching is performed upstream from the deadtime generator so that any abrupt changes that might occur do not violate deadtime on the power stage when in complementary mode.

28.4.2 Functional Details

This section describes the implementation of various sections of the PWM in greater detail.

The following figure is a high-level block diagram of output PWM generation.

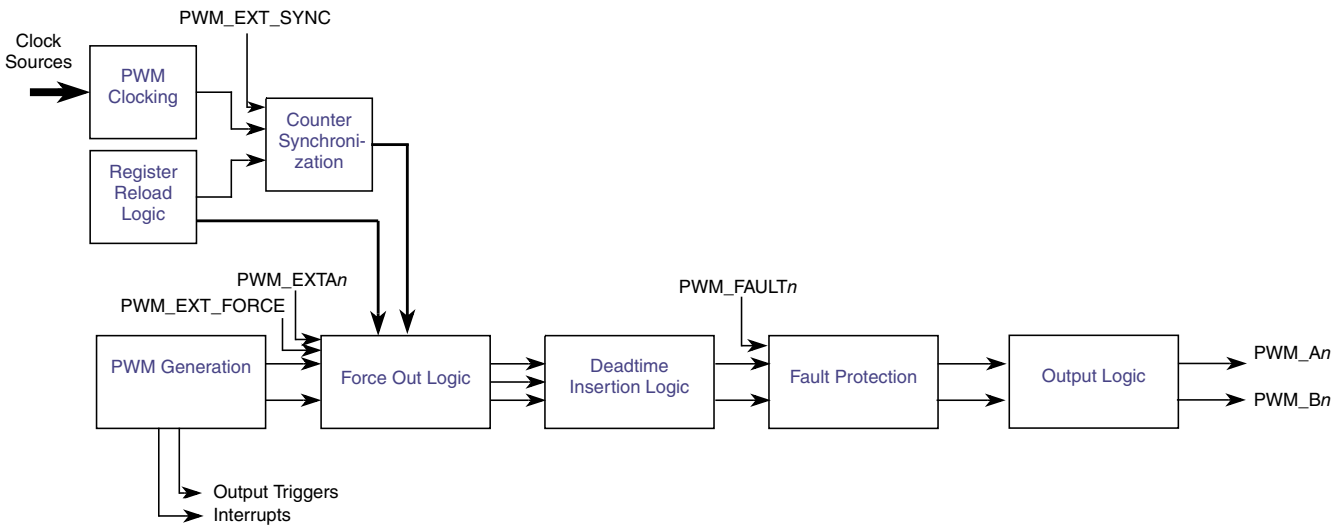


Figure 28-12. High-Level Output PWM Generation Block Diagram

28.4.2.1 PWM Clocking

Figure 28-13 shows the logic used to generate the main counter clock. Each submodule can select between three clock signals: the IPBus clock, EXT_CLK, and AUX_CLK. The EXT_CLK goes to all of the submodules. The AUX_CLK signal is broadcast from submodule0 and can be selected as the clock source by other submodules so that the 8-bit prescaler and MCTRL[RUN] from submodule0 can control all of the submodules.

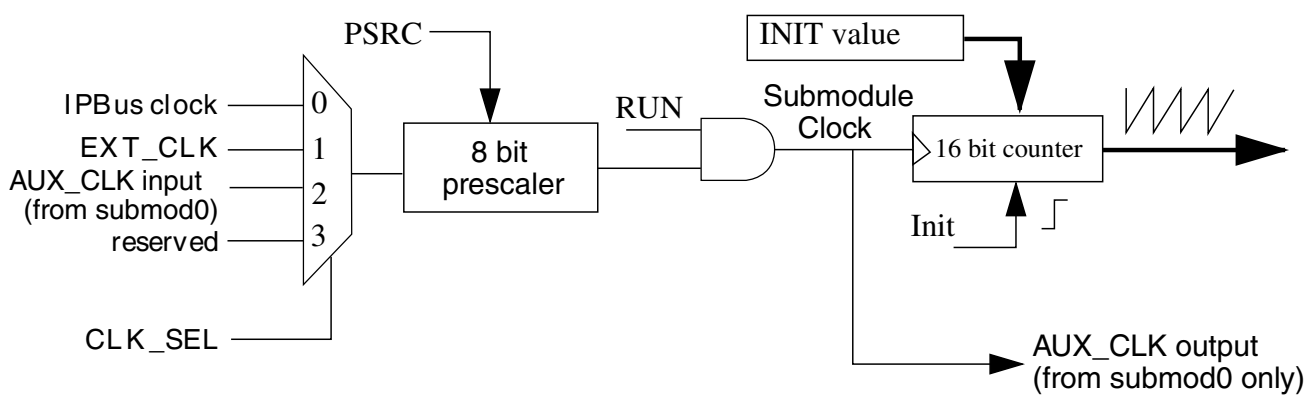


Figure 28-13. Clocking Block Diagram for Each PWM Submodule

To permit lower PWM frequencies, the prescaler produces the PWM clock frequency by dividing the IPBus clock frequency by 1-128. The prescaler bits, CTRL[PRSC], select the prescaler divisor. This prescaler is buffered and will not be used by the PWM generator until MCTRL[LDOK] is set and a new PWM reload cycle begins or CTRL[LDMOD] is set.

28.4.2.2 Register Reload Logic

The register reload logic is used to determine when the outer set of registers for all double buffered register pairs will be transferred to the inner set of registers. The register reload event can be scheduled to occur every "n" PWM cycles using CTRL[LDFQ] and CTRL[FULL]. A half cycle reload option is also supported (CTRL[HALF]) where the reload can take place in the middle of a PWM cycle. The half cycle point is defined by the VAL0 register and does not have to be exactly in the middle of the PWM cycle.

As illustrated in Figure 28-14 the reload signal from submodule0 can be broadcast as the Master Reload signal allowing the reload logic from submodule0 to control the reload of registers in other submodules.

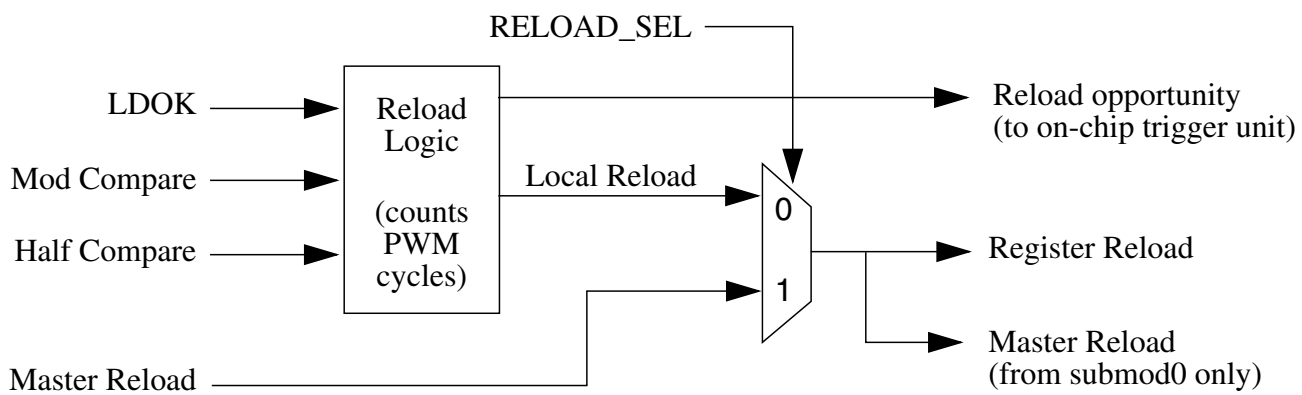


Figure 28-14. Register Reload Logic

28.4.2.3 Counter Synchronization

In the following figure, the 16 bit counter will count up until its output equals VAL1 which is used to specify the counter modulus value. The resulting compare causes a rising edge to occur on the Local Sync signal which is one of four possible sources used to cause the 16 bit counter to be initialized with INIT. If Local Sync is selected as the counter initialization signal, then VAL1 within the submodule effectively controls the timer period (and thus the PWM frequency generated by that submodule) and everything works on a local level.

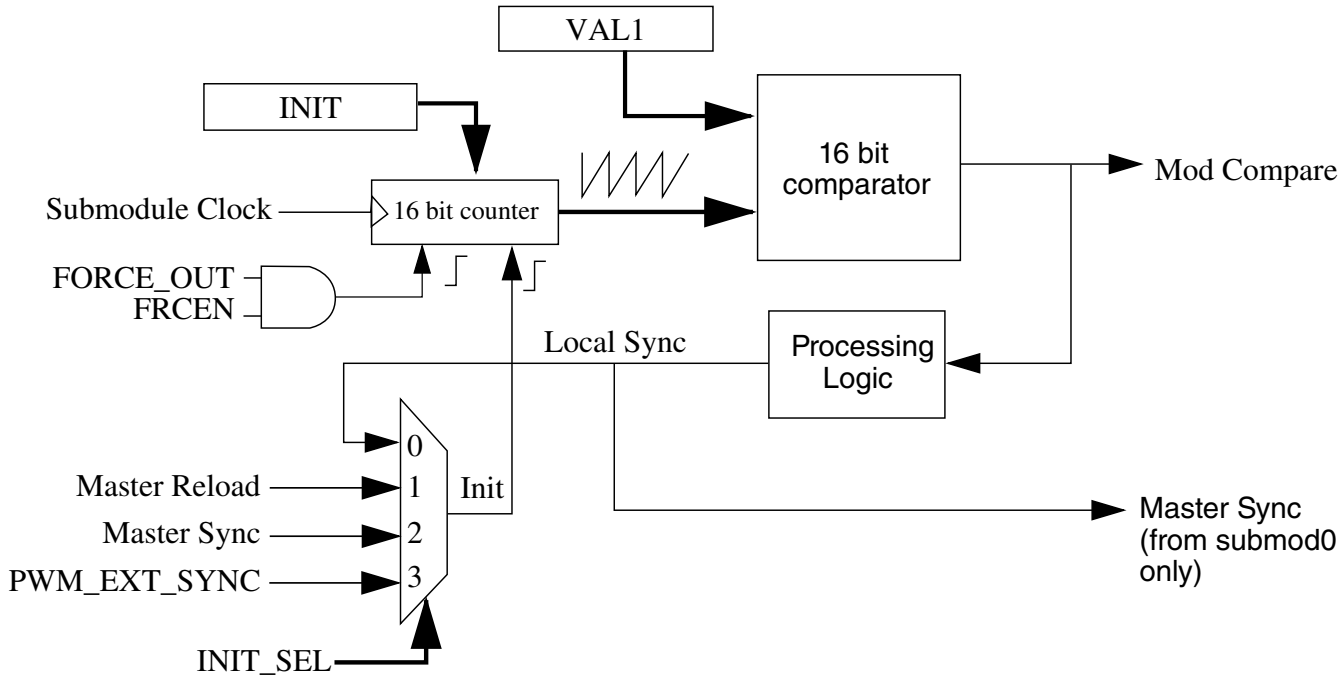


Figure 28-15. Submodule Timer Synchronization

The Master Sync signal originates as the Local Sync from submodule0. If configured to do so, the timer period of any submodule can be locked to the period of the timer in submodule0.

The PWM_EXT_SYNC signal originates on chip or off chip depending on the system architecture. This signal may be selected as the source for counter initialization so that an external source can control the period of all submodules.

If the Master Reload signal is selected as the source for counter initialization, then the period of the counter will be locked to the register reload frequency of submodule0. Since the reload frequency is usually commensurate to the sampling frequency of the software control algorithm, the submodule counter period will therefore equal the sampling period. As a result, this timer can be used to generate output compares or output triggers over the entire sampling period which may consist of several PWM cycles. The Master Reload signal can only originate from submodule0.

The counter can optionally initialize upon the assertion of the FORCE_OUT signal assuming that CTRL2[FRCEN] is set. As indicated by the preceding figure, this constitutes a second init input into the counter which will cause the counter to initialize regardless of which signal is selected as the counter init signal. A forced initialization will also cause a register reload if MCTRL[LDOK] is set.

28.4.2.4 PWM Generation

Figure 28-16 illustrates how PWM generation is accomplished in each submodule. In each case, two comparators and associated VALx registers are utilized for each PWM output signal. One comparator and VALx register are used to control the turn-on edge, while a second comparator and VALx register control the turn-off edge.

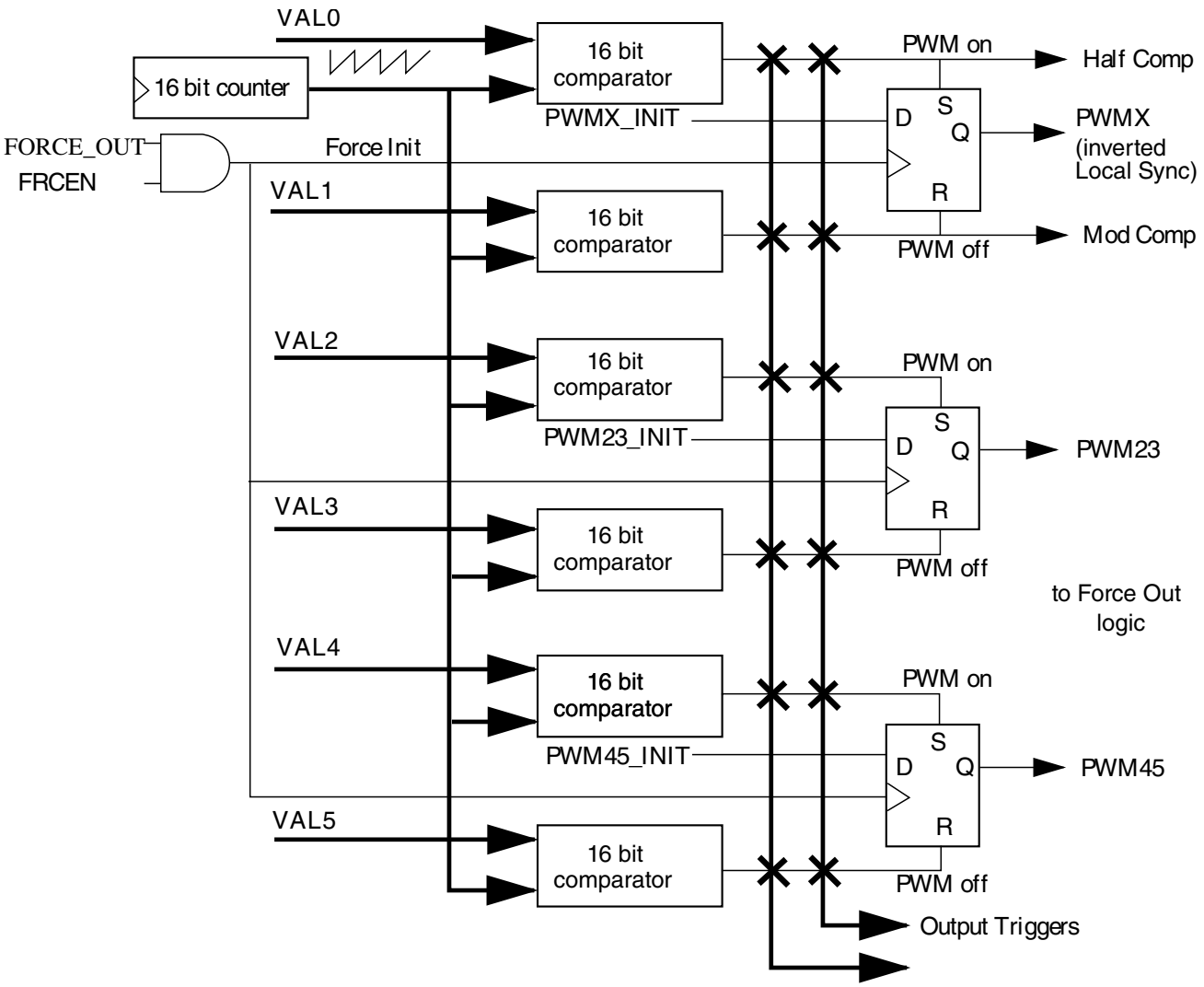


Figure 28-16. PWM Generation Hardware

The generation of the Local Sync signal is performed exactly the same way as the other PWM signals in the submodule. While comparator 0 causes a falling edge of the Local Sync signal, comparator 1 generates a rising edge. Comparator 1 is also hardwired to the reload logic to generate the full cycle reload indicator.

If VAL1 is controlling the modulus of the counter and VAL0 is half of the VAL1 register minus the INIT value, then the half cycle reload pulse will occur exactly half way through the timer count period and the Local Sync will have a 50% duty cycle. On the other hand, if the VAL1 and VAL0 registers are not required for register reloading or

counter initialization, they can be used to modulate the duty cycle of the Local Sync signal, effectively turning it into an auxiliary PWM signal (PWM_X) assuming that the PWM_X pin is not being used for another function such as input capture or deadtime distortion correction. Including the Local Sync signal, each submodule is capable of generating three PWM signals where software has complete control over each edge of each of the signals.

If the comparators and edge value registers are not required for PWM generation, they can also be used for other functions such as output compares, generating output triggers, or generating interrupts at timed intervals.

The 16-bit comparators shown in Figure 28-16are "equal to" comparators. In addition, if both the set and reset of the flip-flop are asserted, then the flop output goes to 0.

28.4.2.5 Output Compare Capabilities

By using the VALx registers in conjunction with the submodule timer and 16 bit comparators, buffered output compare functionality can be achieved with no additional hardware required. Specifically, the following output compare functions are possible:

- An output compare sets the output high
- An output compare sets the output low
- An output compare generates an interrupt
- An output compare generates an output trigger

In PWM generation, an output compare is initiated by programming a VALx register for a timer compare, which in turn causes the output of the D flip-flop to either set or reset. For example, if an output compare is desired on the PWM_A signal that sets it high, VAL2 would be programmed with the counter value where the output compare should take place. However, to prevent the D flip-flop from being reset again after the compare has occurred, the VAL3 register must be programmed to a value outside of the modulus range of the counter. Therefore, a compare that would result in resetting the D flip-flop output would never occur. Conversely, if an output compare is desired on the PWM_A signal that sets it low, the VAL3 register is programmed with the appropriate count value and the VAL2 register is programmed with a value outside the counter modulus range. Regardless of whether a high compare or low compare is programmed, an interrupt or output trigger can be generated when the compare event occurs.

28.4.2.6 Force Out Logic

For each submodule, software can select between eight signal sources for the FORCE_OUT signal: local CTRL2[FORCE], the Master Force signal from submodule0, the local Reload signal, the Master Reload signal from submodule0, the Local Sync signal, the Master Sync signal from submodule0, the EXT_SYNC signal from on- or off-chip, or the EXT_FORCE signal from on- or off-chip depending on the chip architecture. The local signals are used when the user simply wants to change the signals on the output pins of the submodule without regard for synchronization with other submodules. However, if it is required that all signals on all submodule outputs change at the same time, the Master, EXT_SYNC, or EXT_FORCE signals should be selected.

Figure 28-17 illustrates the Force logic. The SEL23 and SEL45 fields each choose from one of four signals that can be supplied to the submodule outputs: the PWM signal, the inverted PWM signal, a binary level specified by software via the OUT23 and OUT45 bits, or the PWM_EXTA or PWM_EXTB alternate external control signals. The selection can be determined ahead of time and, when a FORCE_OUT event occurs, these values are presented to the signal selection mux that immediately switches the requested signal to the output of the mux for further processing downstream.

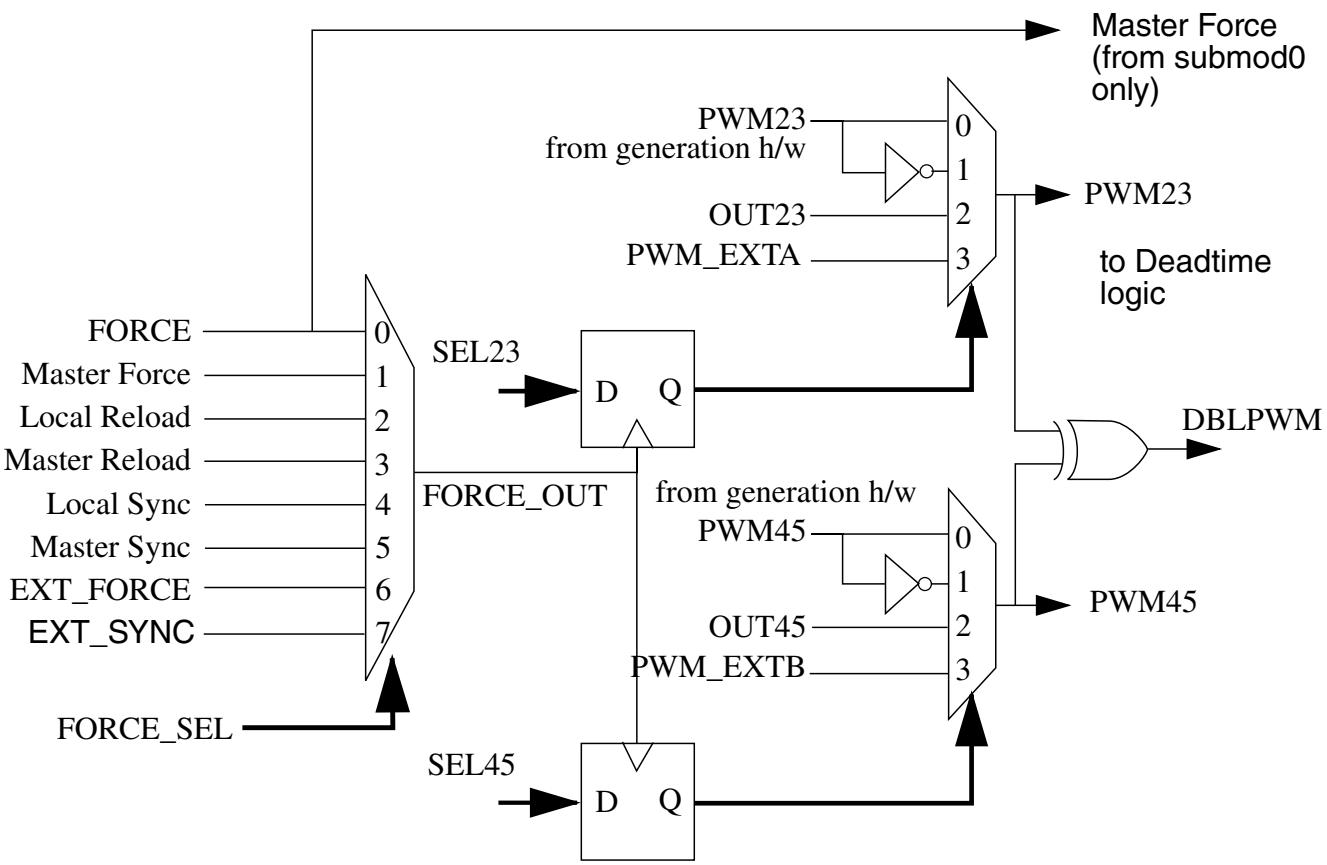


Figure 28-17. Force Out Logic

The local CTRL2[FORCE] signal of submodule0 can be broadcast as the Master Force signal to other submodules. This feature allows the CTRL2[FORCE] of submodule0 to synchronously update all of the submodule outputs at the same time. The EXT_FORCE signal originates from outside the PWM module from a source such as a timer or digital comparators in the Analog-to-Digital Converter.

28.4.2.7 Independent or Complementary Channel Operation

Writing a logic one to CTRL2[INDEP] configures the pair of PWM outputs as two independent PWM channels. Each PWM output is controlled by its own VALx pair operating independently of the other output.

Writing a logic zero to CTRL2[INDEP] configures the PWM output as a pair of complementary channels. The PWM pins are paired as shown in Figure 28-18 in complementary channel operation. Which signal is connected to the output pin (PWM23 or PWM45) is determined by MCTRL[IPOL].

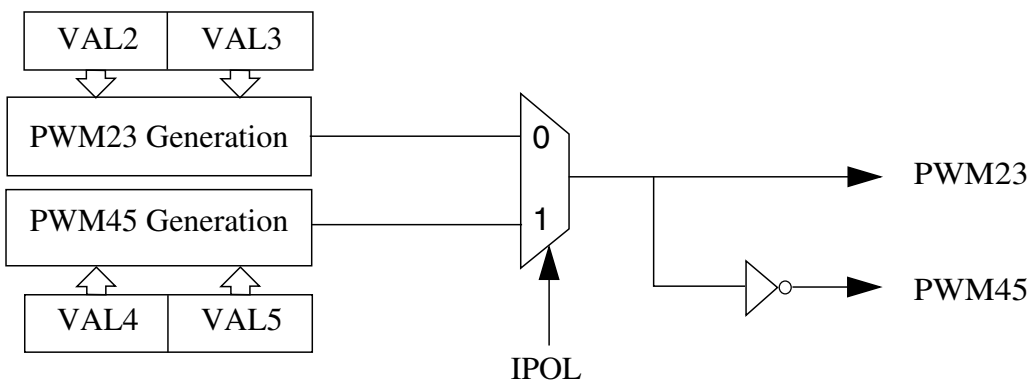


Figure 28-18. Complementary Channel Pair

Complementary operation allows the use of the deadtime insertion feature.

28.4.2.8 Deadtime Insertion Logic

The following figure shows the deadtime insertion logic of each submodule which is used to create non-overlapping complementary signals when not in independent mode.

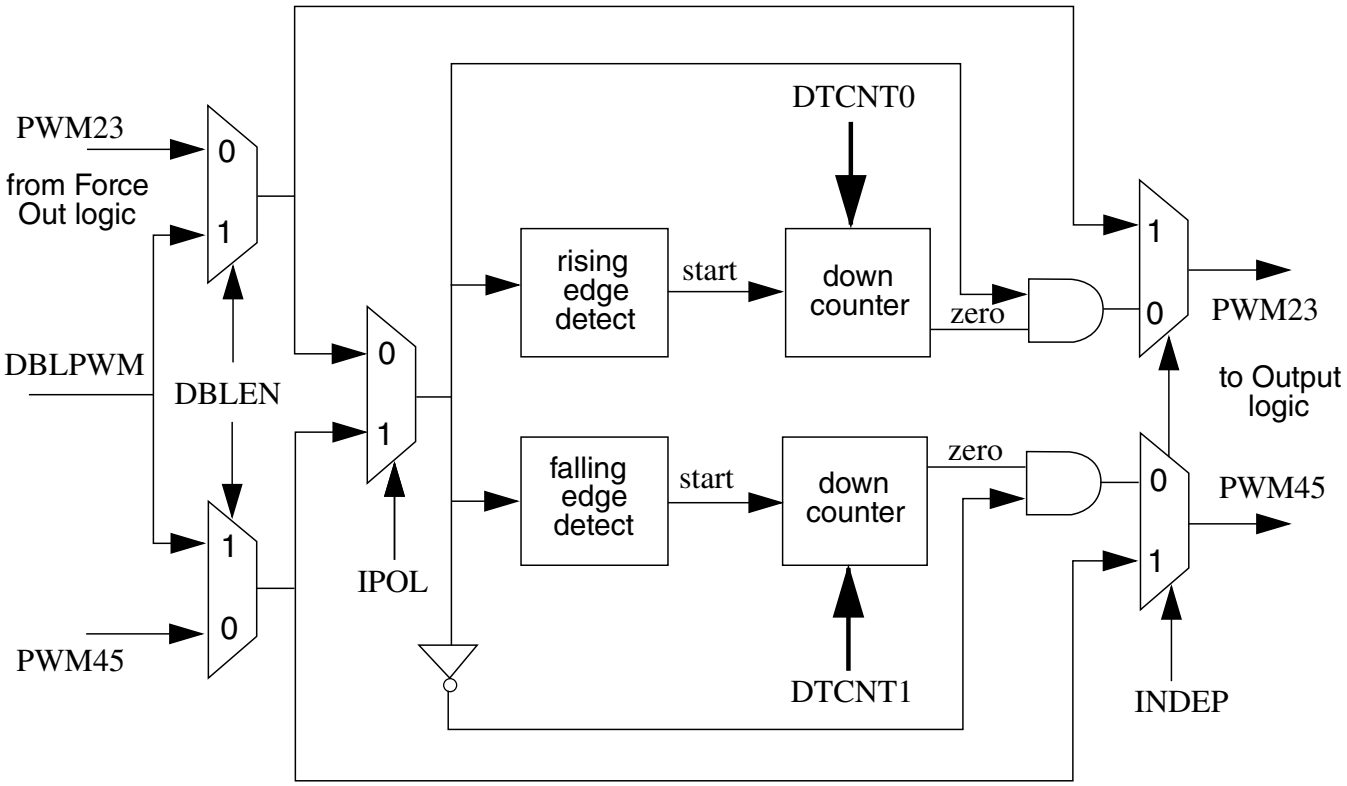


Figure 28-19. Deadtime Insertion Logic

While in the complementary mode, a PWM pair can be used to drive top/bottom transistors, as shown in the figure. When the top PWM channel is active, the bottom PWM channel is inactive, and vice versa.

Note

To avoid short circuiting the DC bus and endangering the transistor, there must be no overlap of conducting intervals between top and bottom transistor. But the transistor's characteristics may make its switching-off time longer than switching-on time. To avoid the conducting overlap of top and bottom transistors, deadtime needs to be inserted in the switching period, as illustrated in the following figure.

The deadtime generators automatically insert software-selectable activation delays into the pair of PWM outputs. The deadtime registers (DTCNT0 and DTCNT1) specify the number of IPBus clock cycles to use for deadtime delay. Every time the deadtime generator inputs change state, deadtime is inserted. Deadtime forces both PWM outputs in the pair to the inactive state.

When deadtime is inserted in complementary PWM signals connected to an inverter driving an inductive load, the PWM waveform on the inverter output will have a different duty cycle than what appears on the output pins of the PWM module. This results in a distortion in the voltage applied to the load. A method of correcting this, adding to or subtracting from the PWM value used, is discussed next.

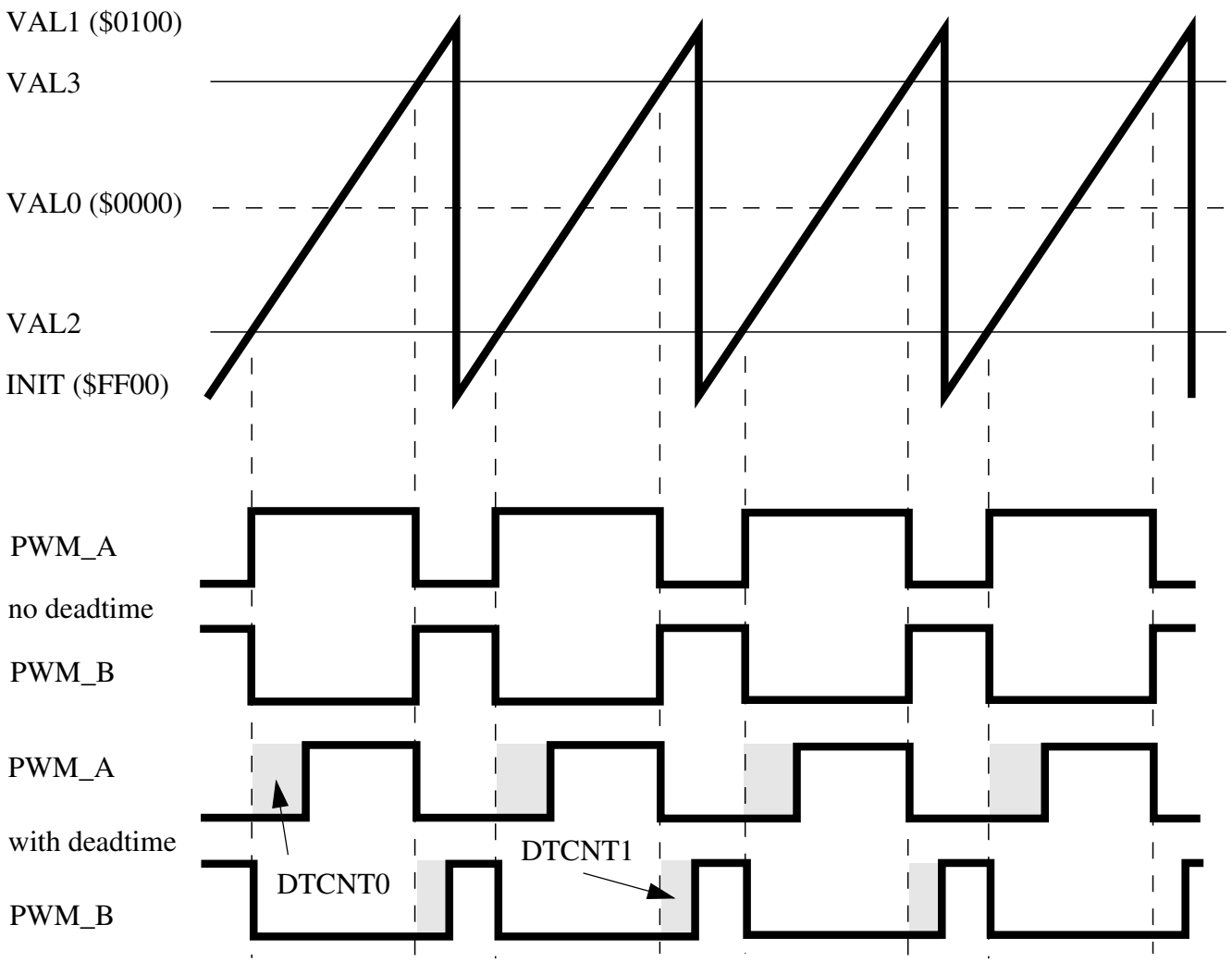


Figure 28-20. Deadtime Insertion

28.4.2.8.1 Top/Bottom Correction

In complementary mode, either the top or the bottom transistor controls the output voltage. However, deadtime has to be inserted to avoid overlap of conducting interval between the top and bottom transistor. Both transistors in complementary mode are off during deadtime, allowing the output voltage to be determined by the current status of load and introduce distortion in the output voltage. See the following figure.

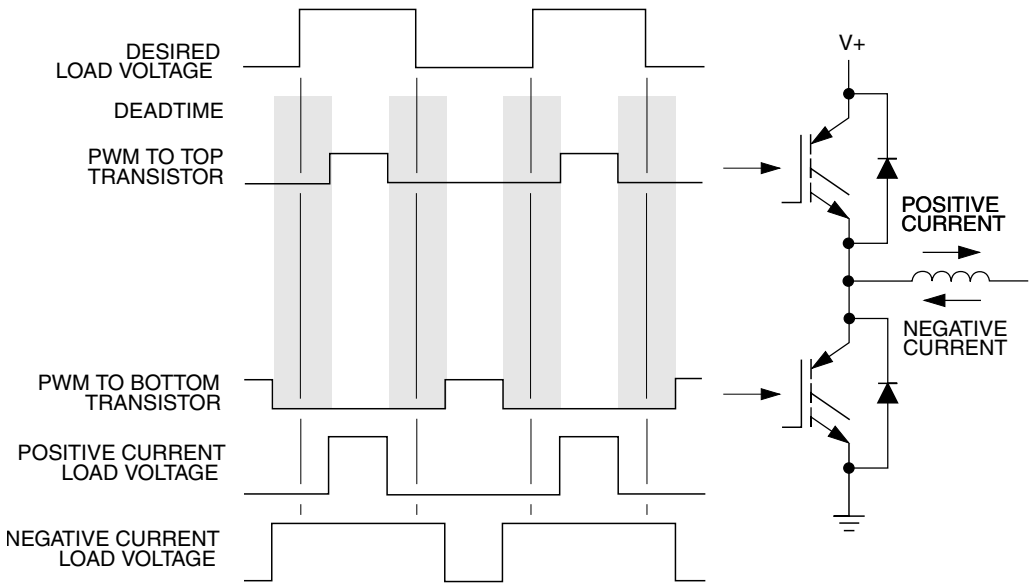


Figure 28-21. Deadtime Distortion

During deadtime, load inductance distorts output voltage by keeping current flowing through the diodes. This deadtime current flow creates a load voltage that varies with current direction. With a positive current flow, the load voltage during deadtime is equal to the bottom supply, putting the top transistor in control. With a negative current flow, the load voltage during deadtime is equal to the top supply putting the bottom transistor in control.

Remembering that the original PWM pulse widths where shortened by deadtime insertion, the averaged sinusoidal output will be less than the desired value. However, when deadtime is inserted, it creates a distortion in the inverter outputs. This distortion is aggravated by dissimilar turn-on and turn-off delays of each of the transistors. By giving the PWM module information on which transistor is controlling at a given time this distortion can be corrected.

For a typical circuit in complementary channel operation, only one of the transistors will be effective in controlling the output voltage at any given time. This depends on the direction of the inverter current for that pair, as the preceding figure shows. To correct distortion one of two different factors must be added to the desired PWM value, depending on whether the top or bottom transistor is controlling the output voltage. Therefore, the software is responsible for calculating both compensated PWM values prior to placing them in the VALx registers. Either the VAL2/VAL3 or the VAL4/VAL5 register pair controls the pulse width at any given time. For a given PWM pair, whether the VAL2/VAL3 or VAL4/VAL5 pair is active depends on either:

- The state of the current status pin, PWMX, for that driver
- The state of the odd/even correction bit, MCTRL[IPOL], for that driver

To correct deadtime distortion, software can decrease or increase the value in the appropriate VALx register.

- In edge-aligned operation, decreasing or increasing the PWM value by a correction value equal to the deadtime typically compensates for deadtime distortion.
- In center-aligned operation, decreasing or increasing the PWM value by a correction value equal to one-half the deadtime typically compensates for deadtime distortion.

28.4.2.8.2 Manual Correction

To detect the current status, the voltage on each PWMX pin is sampled twice in a PWM period, at the end of each deadtime. The value is stored in CTRL[DT]. CTRL[DT] is a timing marker especially indicating when to toggle between PWM value registers. Software can then set MCTRL[IPOL] to switch between VAL2/VAL3 and VAL4/VAL5 register pairs according to CTRL[DT] values.

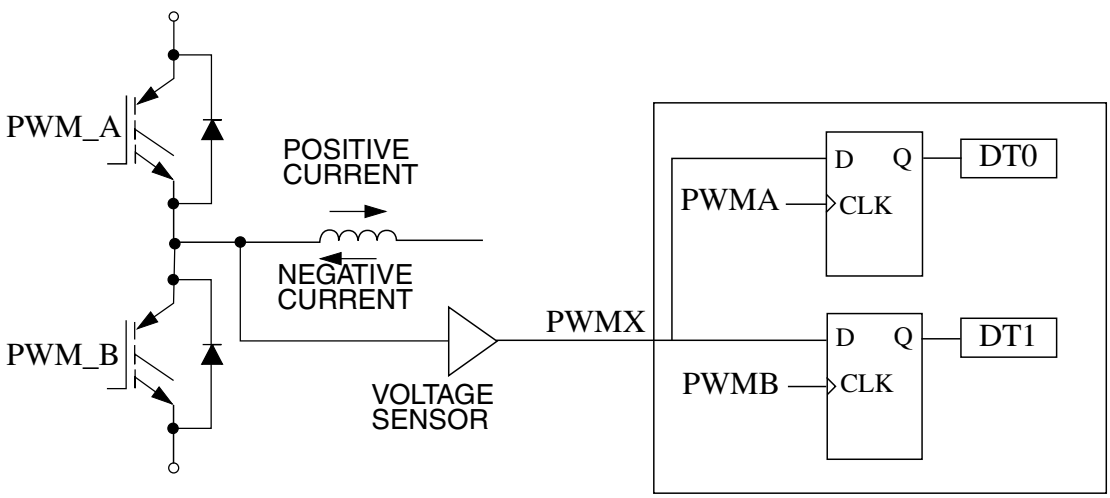


Figure 28-22. Current-status Sense Scheme for Deadtime Correction

Both D flip-flops latch low, CTRL[DT] = 00, during deadtime periods if current is large and flowing out of the complementary circuit. See the preceding figure. Both D flip-flops latch the high, CTRL[DT] = 11, during deadtime periods if current is also large and flowing into the complementary circuit.

However, under low-current, the output voltage of the complementary circuit during deadtime is somewhere between the high and low levels. The current cannot free-wheel through the opposition anti-body diode, regardless of polarity, giving additional distortion when the current crosses zero. **Sampled results will be CTRL[DT] = b10. Thus, the best time to change one PWM value register to another is just before the current zero crossing.**

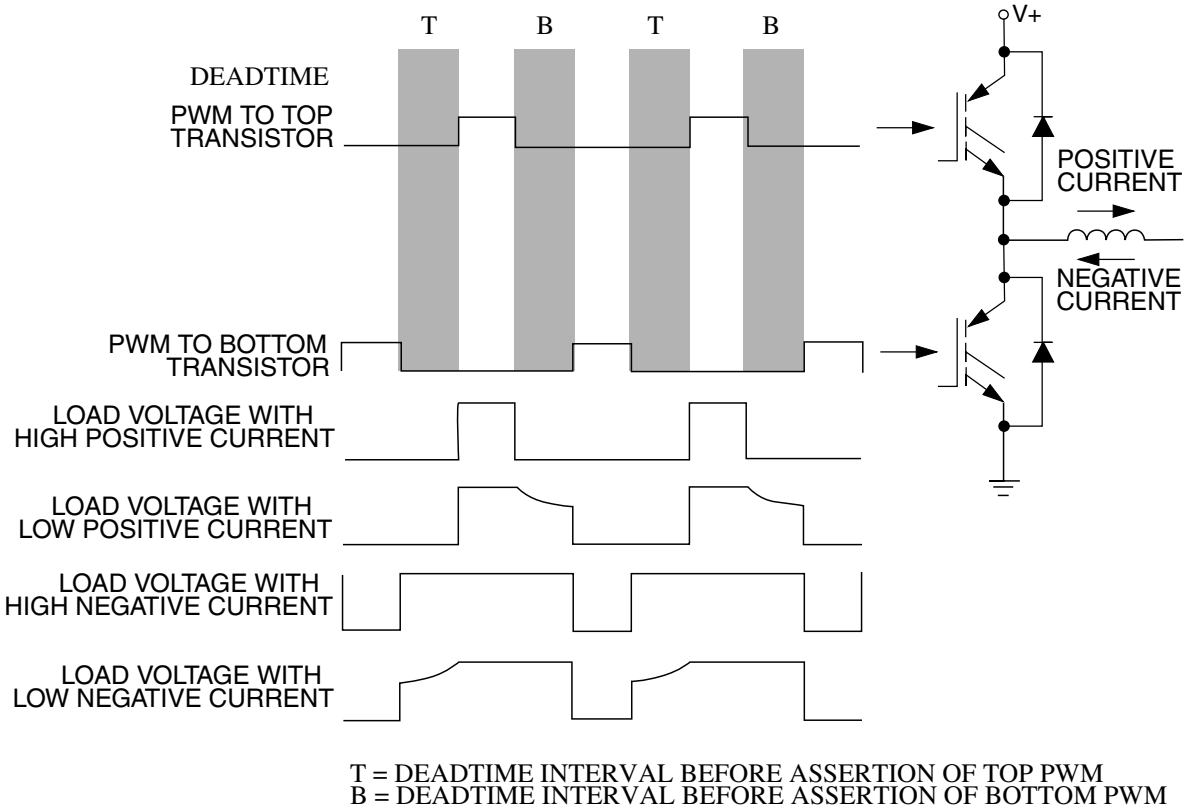


Figure 28-23. Output Voltage Waveforms

28.4.2.9 Fractional Delay Logic

For applications where more resolution than a single IPBus clock period is needed, the fractional delay logic can be used to achieve fine resolution on the rising and falling edges of the PWM_A and PWM_B outputs and fine resolution for the PWM period. Enable the use of the fractional delay logic by setting FRCTRL[FRACx_EN]. The FRACVALx registers act as a fractional clock cycle addition to the turn on and turn off count specified by the VAL2, VAL3, VAL4, or VAL5 registers. The FRACVAL1 register acts as a fractional increase in the PWM period as defined by VAL1. If FRACVAL1 is programmed to a non-zero value, then the largest value for the VAL1 register is 0xFFFE for unsigned usage or 0x7FFE for signed usage. This limit is needed in order to avoid counter rollovers when accumulating the fractional additional period.

The results of the fractional delay logic depend on whether or not the PWM submodule has an analog micro-edge placer block available.

28.4.2.9.1 Fractional Delay Logic with Micro-Edge Placement Block

Using the micro-edge placer block requires that the IPBus clock to the PWM be set at a defined frequency. The micro-edge placer is powered up by setting FRCTRL[FRAC_PU]. Enable fine edge control on the various PWM edges by setting FRCTRL[FRACx_EN]. The fractional values in the FRACVALx registers allow placing the PWM edge or PWM period to a granularity of 1/32 of the IPBus clock period. For example, if you desire the rising edge of the PWMA output to occur at a count of 12.25, then program VAL2 with 0x000C and FRACVAL2 with 0x4000. Using FRACVAL1 will adjust the PWM period with the same granularity of 1/32 of a clock period.

If the FRCTRL[FRAC_PU] bits in all of the submodules are clear, then the micro-edge placer is powered down, and alternate clock frequencies can be used without the micro-edge placement feature.

28.4.2.10 Output Logic

The following figure shows the output logic of each submodule including how each PWM output has individual fault disabling, polarity control, and output enable. This allows for maximum flexibility when interfacing to the external circuitry.

The PWM23 and PWM45 signals which are output from the deadtime logic (refer to the figure) are positive true signals. In other words, a high level on these signals should result in the corresponding transistor in the PWM inverter being turned ON. The voltage level required at the PWM output pin to turn the transistor ON or OFF is a function of the logic between the pin and the transistor. Therefore, it is imperative that the user program OCTRL[POLA] and OCTRL[POLB] before enabling the output pins. A fault condition can result in the PWM output being tristated, forced to a logic 1, or forced to a logic 0 depending on the values programmed into the OCTRL[PWMxFS] fields.

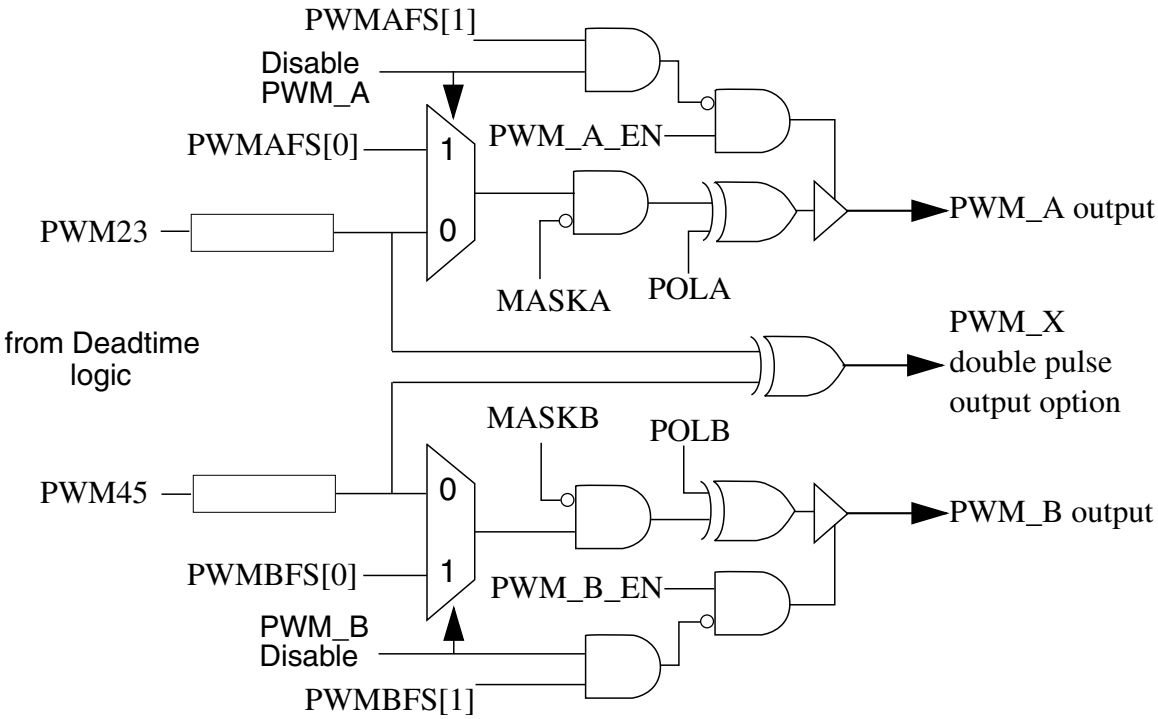


Figure 28-24. Output Logic

28.4.2.11 E-Capture

Commensurate with the idea of controlling both edges of an output signal, the Enhanced Capture (E-Capture) logic is designed to measure both edges of an input signal. As a result, when a submodule pin is configured for input capture, the CVALx registers associated with that pin are used to record the edge values.

The following figure is a block diagram of the E-Capture circuit. Upon entering the pin input, the signal is split into two paths. One goes straight to a mux input where software can select to pass the signal directly to the capture logic for processing. The other path connects the signal to an 8 bit counter which counts both the rising and falling edges of the signal. The output of this counter is compared to an 8 bit value that is specified by the user (EDGCMPx) and when the two values are equal, the comparator generates a pulse that resets the counter. This pulse is also supplied to the mux input where software can select it to be processed by the capture logic. This feature permits the E-Capture circuit to count up to 256 edge events before initiating a capture event. this feature is useful for dividing down high frequency signals for capture processing so that capture interrupts don't overwhelm the CPU. Also, this feature can be used to simply generate an interrupt after "n" events have been counted.

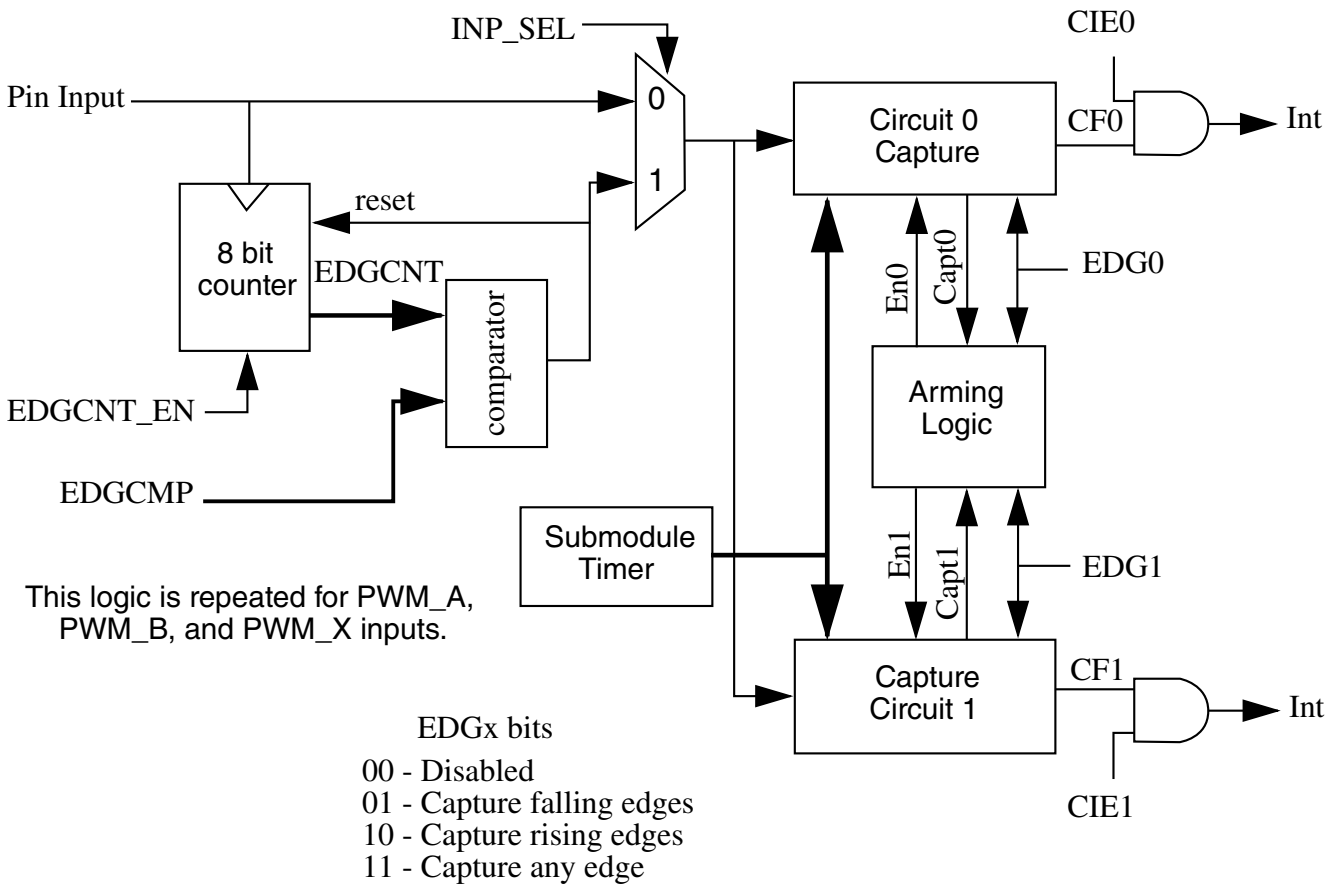


Figure 28-25. Enhanced Capture (E-Capture) Logic

Based on the mode selection, the mux selects either the pin input or the compare output from the count/compare circuit to be processed by the capture logic. The selected signal is routed to two separate capture circuits which work in tandem to capture sequential edges of the signal. The type of edge to be captured by each circuit is determined by CAPTCTRLx[EDGx1] and CAPTCTRLx[EDGx0], whose functionality is listed in the preceding figure. Also, controlling the operation of the capture circuits is the arming logic which allows captures to be performed in a free running (continuous) or one shot fashion. In free running mode, the capture sequences will be performed indefinitely. If both capture circuits are enabled, they will work together in a ping-pong style where a capture event from one circuit leads to the arming of the other and vice versa. In one shot mode, only one capture sequence will be performed. If both capture circuits are enabled, capture circuit 0 is first armed and when a capture event occurs, capture circuit 1 is armed. Once the second capture occurs, further captures are disabled until another capture sequence is initiated. Both capture circuits are also capable of generating an interrupt to the CPU.

28.4.2.12 Fault Protection

Fault protection can control any combination of PWM output pins. Faults are generated by a logic one on any of the FAULTx pins. This polarity can be changed via FCTRL[FLVL]. Each FAULTx pin can be mapped arbitrarily to any of the PWM outputs. When fault protection hardware disables PWM outputs, the PWM generator continues to run, only the output pins are forced to logic 0, logic 1, or tristated depending the values of OCTRL[PWMxFS].

The fault decoder disables PWM pins selected by the fault logic and the disable mapping (DISMAPn) registers. The following figure shows an example of the fault disable logic. Each bank of bits in DISMAPn control the mapping for a single PWM pin. See the following table.

The fault protection is enabled even when the PWM module is not enabled; therefore, a fault will be latched in and must be cleared in order to prevent an interrupt when the PWM is enabled.

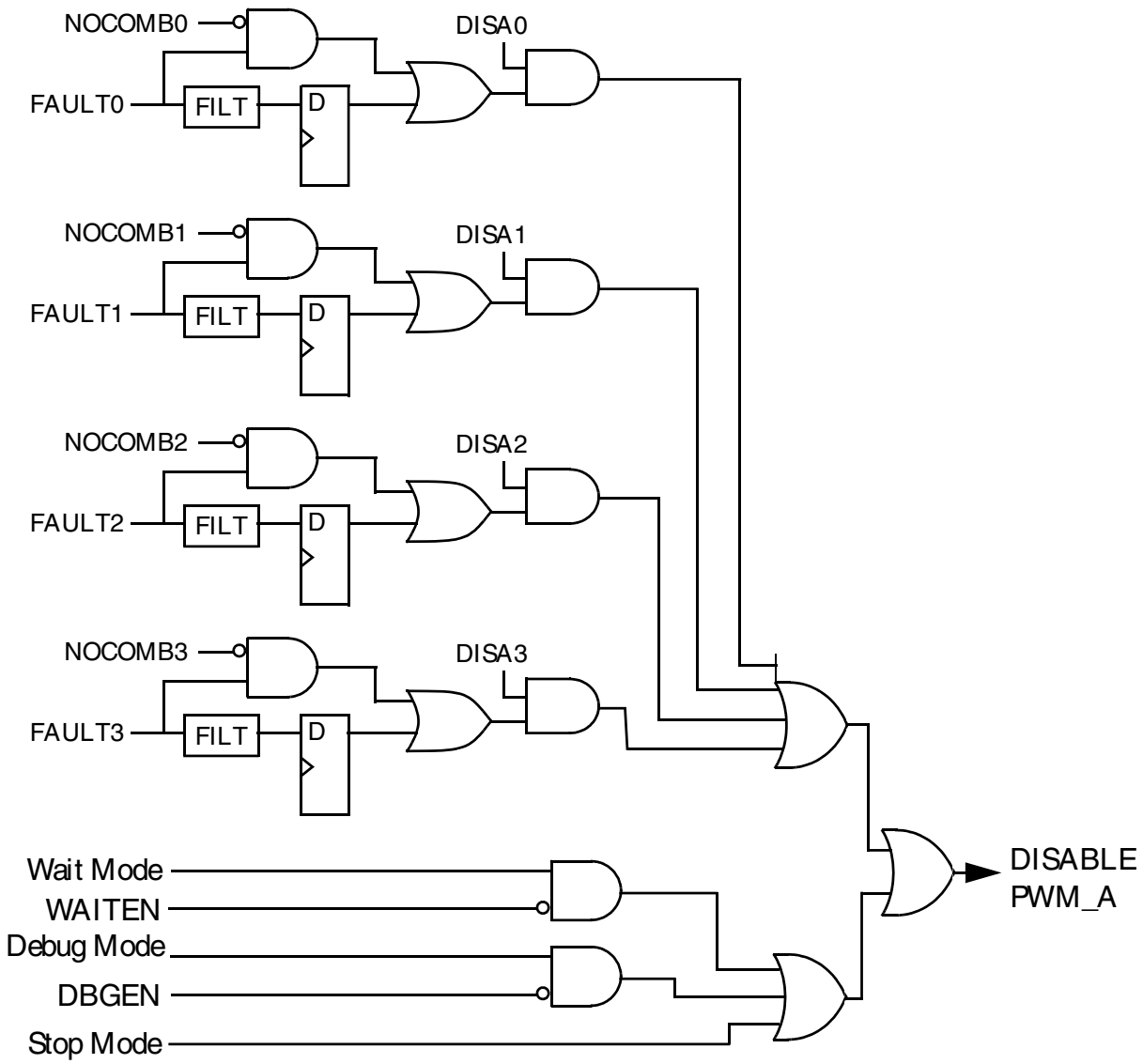


Figure 28-26. Fault Decoder for PWM_A

Table 28-2. Fault Mapping

PWM Pin	Controlling Register Bits
PWM_A	DISMAP0[DIS0A]
PWM_B	DISMAP0[DIS0B]
PWM_X	DISMAP0[DIS0X]

28.4.2.12.1 Fault Pin Filter

Each fault pin has a programmable filter that can be bypassed. The sampling period of the filter can be adjusted with FFILT[FILT_PER]. The number of consecutive samples that must agree before an input transition is recognized can be adjusted using FFILT[FILT_CNT]. Setting FFILT[FILT_PER] to all 0 disables the input filter for a given FAULTx pin.

Upon detecting a logic 0 on the filtered FAULTx pin (or a logic 1 if FCTRL[FLVLx] is set), the corresponding FSTS[FFPINx] and fault flag, FSTS[FFLAGx], bits are set. FSTS[FFPINx] remains set as long as the filtered FAULTx pin is zero. Clear FSTS[FFLAGx] by writing a logic 1 to FSTS[FFLAGx].

If the FIEx, FAULTx pin interrupt enable bit is set, FSTS[FFLAGx] generates a CPU interrupt request. The interrupt request latch remains set until:

- Software clears FSTS[FFLAGx] by writing a logic one to the bit
- Software clears the FIEx bit by writing a logic zero to it
- A reset occurs

Even with the filter enabled, there is a combinational path from the FAULTx inputs to the PWM pins. This logic is also capable of holding a fault condition in the event of loss of clock to the PWM module.

28.4.2.12.2 Automatic Fault Clearing

Setting an automatic clearing mode bit, FCTRL[FAUTOx], configures faults from the FAULTx pin for automatic clearing.

When FCTRL[FAUTOx] is set, disabled PWM pins are enabled when the FAULTx pin returns to logic one and a new PWM full or half cycle begins. See the following figure. If FSTS[FFULLx] is set, then the disabled PWM pins are enabled at the start of a full cycle. If FSTS[FHALFx] is set, then the disabled PWM pins are enabled at the start of a half cycle. Clearing FSTS[FFLAGx] does not affect disabled PWM pins when FCTRL[FAUTOx] is set.

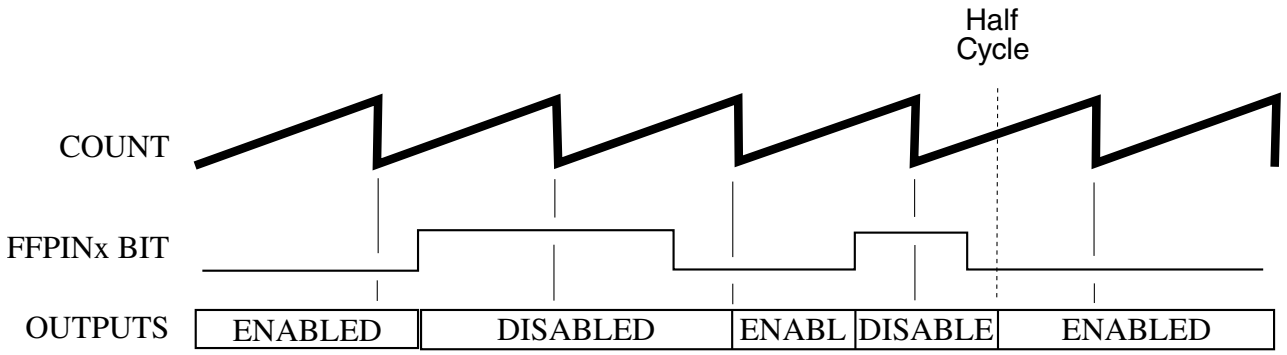


Figure 28-27. Automatic Fault Clearing

28.4.2.12.3 Manual Fault Clearing

Clearing the automatic clearing mode bit, FCTRL[FAUTOx], configures faults from the FAULTx pin for manual clearing:

- If the fault safety mode bits, FCTRL[FSAFEx], are clear, then PWM pins disabled by the FAULTx pins are enabled when:
  - Software clears the corresponding FSTS[FFLAGx] flag
  - The pins are enabled when the next PWM full or half cycle begins regardless of the logic level detected by the filter at the FAULTx pin. See the first following figure. If FSTS[FFULLx] is set, then the disabled PWM pins are enabled at the start of a full cycle. If FSTS[FHALFx] is set, then the disabled PWM pins are enabled at the start of a half cycle.
- If the fault safety mode bits, FCTRL[FSAFEx], are set, then PWM pins disabled by the FAULTx pins are enabled when:
  - Software clears the corresponding FSTS[FFLAGx] flag
  - The filter detects a logic one on the FAULTx pin at the start of the next PWM full or half cycle boundary. See the second following figure. If FSTS[FFULLx] is set, then the disabled PWM pins are enabled at the start of a full cycle. If FSTS[FHALFx] is set, then the disabled PWM pins are enabled at the start of a half cycle.

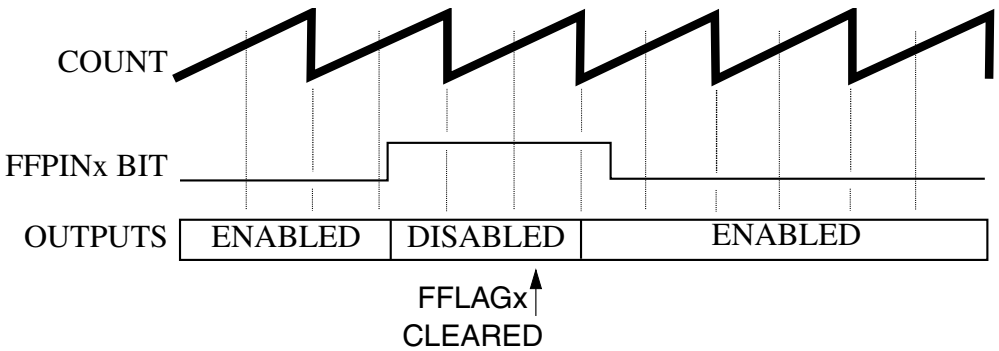


Figure 28-28. Manual Fault Clearing (FCTRL[FSAFE]=0)

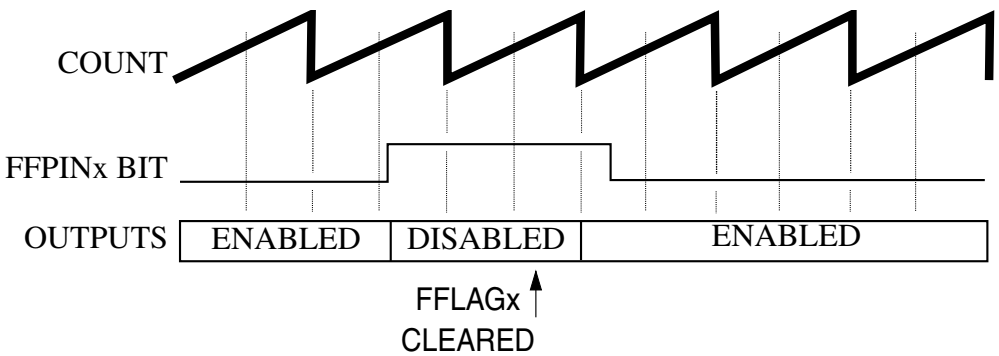


Figure 28-29. Manual Fault Clearing (FCTRL[FSAFE]=1)

Note

Fault protection also applies during software output control when the SEL23 and SEL45 fields are set to select OUT23 and OUT45 bits or PWM_EXTA and PWM_EXTB. Fault clearing still occurs at half PWM cycle boundaries while the PWM generator is engaged, MCTRL[RUN] equals one. But the OUTx bits can control the PWM pins while the PWM generator is off, MCTRL[RUN] equals zero. Thus, fault clearing occurs at IPBus cycles while the PWM generator is off and at the start of PWM cycles when the generator is engaged.

28.4.2.12.4 Fault Testing

FTST[FTEST] is used to simulate a fault condition on each of the fault inputs within that fault channel.

28.4.3 PWM Generator Loading

28.4.3.1 Load Enable

MCTRL[LDOK] enables loading of the following PWM generator parameters:

- The prescaler divisor—from CTRL[PRSC]
- The PWM period and pulse width—from the INIT and VALx registers

MCTRL[LDOK] allows software to finish calculating all of these PWM parameters so they can be synchronously updated. The CTRL[PRSC], INIT, and VALx registers are loaded by software into a set of outer buffers. When MCTRL[LDOK] is set, these values are transferred to an inner set of registers at the beginning of the next PWM reload cycle to be used by the PWM generator. These values can be transfered to the inner set of registers immediately upon setting MCTRL[LDOK] if CTRL[LDMOD] is set. Set MCTRL[LDOK] by reading it when it is a logic zero and then writing a logic one to it. After loading, MCTRL[LDOK] is automatically cleared.

28.4.3.2 Load Frequency

CTRL[LDFQ] selects an integral loading frequency of one to 16 PWM reload opportunities. CTRL[LDFQ] takes effect at every PWM reload opportunity, regardless the state of MCTRL[LDOK]. CTRL[HALF] and CTRL[FULL] control reload timing. If CTRL[FULL] is set, a reload opportunity occurs at the end of every PWM cycle when the count equals VAL1. If CTRL[HALF] is set, a reload opportunity occurs at the half cycle when the count equals VAL0. If both CTRL[HALF] and CTRL[FULL] are set, a reload opportunity occurs twice per PWM cycle when the count equals VAL1 and when it equals VAL0.

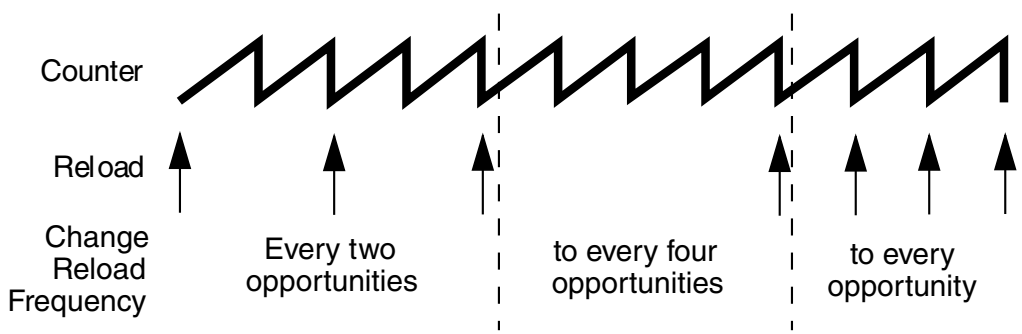


Figure 28-30. Full Cycle Reload Frequency Change

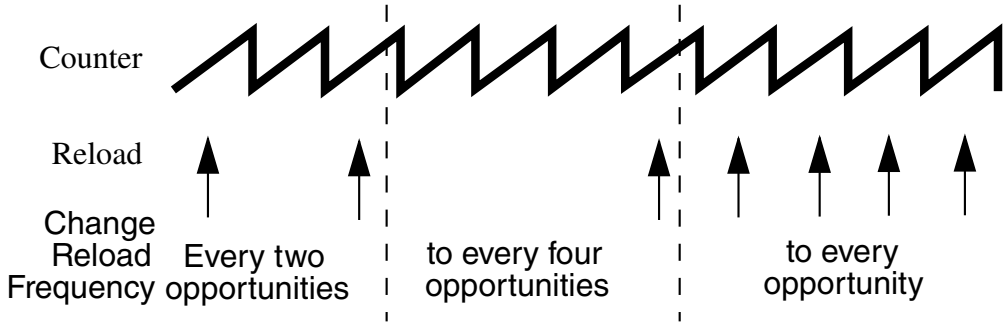


Figure 28-31. Half Cycle Reload Frequency Change

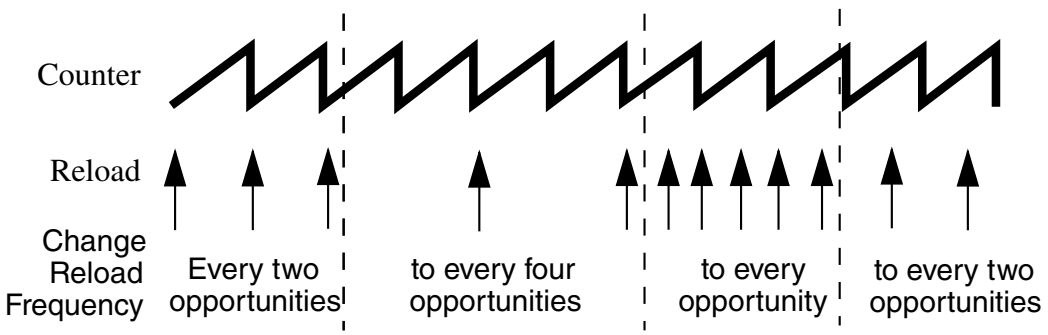


Figure 28-32. Full and Half Cycle Reload Frequency Change

28.4.3.3 Reload Flag

At every reload opportunity the PWM Reload Flag (STS[RF]) is set. Setting STS[RF] happens even if an actual reload is prevented by MCTRL[LDOK]. If the PWM reload interrupt enable bit, INTEN[RIE], is set, the STS[RF] flag generates CPU interrupt requests allowing software to calculate new PWM parameters in real time. When INTEN[RIE] is not set, reloads still occur at the selected reload rate without generating CPU interrupt requests.

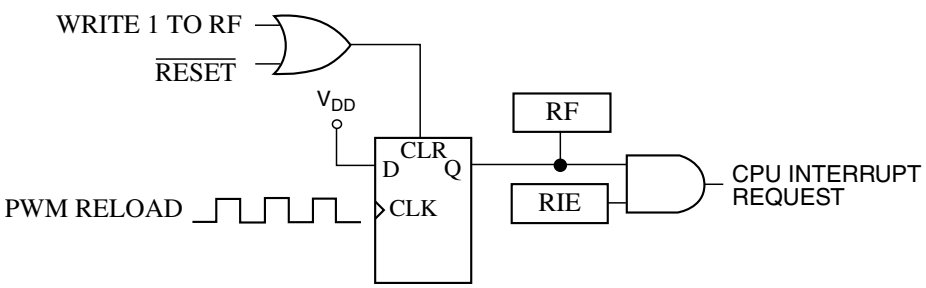


Figure 28-33. PWMF Reload Interrupt Request

28.4.3.4 Reload Errors

Whenever one of the VALx, FRACVALx, or CTRL[PRSC] registers is updated, the STS[RUF] flag is set to indicate that the data is not coherent. STS[RUF] will be cleared by a successful reload which consists of the reload signal while MCTRL[LDOK] is set. If STS[RUF] is set and MCTRL[LDOK] is clear when the reload signal occurs, a reload error has taken place and STS[REF] is set. If STS[RUF] is clear when a reload signal asserts, then the data is coherent and no error will be flagged.

28.4.3.5 Initialization

Initialize all registers and set MCTRL[LDOK] before setting MCTRL[RUN].

Note

Even if MCTRL[LDOK] is not set, setting MCTRL[RUN] also sets the STS[RF] flag. To prevent a CPU interrupt request, clear INTEN[RIE] before setting MCTRL[RUN].

The PWM generator uses the last values loaded if MCTRL[RUN] is cleared and then set while MCTRL[LDOK] equals zero.

When MCTRL[RUN] is cleared:

- The STS[RF] flag and pending CPU interrupt requests are not cleared
- All fault circuitry remains active
- Software/external output control remains active
- Deadtime insertion continues during software/external output control

28.5 Resets

All PWM registers are reset to their default values upon any system reset.

The reset forces all registers to their reset states and tri-states the PWM outputs.

28.6 Interrupts

Each of the submodules within the eFlexPWM module can generate an interrupt from several sources. The fault logic can also generate interrupts. The interrupt service routine (ISR) must check the related interrupt enables and interrupt flags to determine the actual cause of the interrupt.

Table 28-3. Interrupt Summary

Core Interrupt	Interrupt Flag	Interrupt Enable	Name	Description
PWM_CMP0	SM0STS[CMPF]	SM0INTEN[CMPIE]	Submodule 0 compare interrupt	Compare event has occurred
PWM_CAP0	SM0STS[CFA1], SM0STS[CFA0], SM0STS[CFB1], SM0STS[CFB0], SM0STS[CFX1], SM0STS[CFX0]	SM0INTEN[CFA1IE], SM0INTEN[CFA0IE], SM0INTEN[CFB1IE], SM0INTEN[CFB0IE], SM0INTEN[CFX1IE], SM0INTEN[CFX0IE]	Submodule 0 input capture interrupt	Input capture event has occurred
PWM_RELOAD0	SM0STS[RF]	SM0INTEN[RIE]	Submodule 0 reload interrupt	Reload event has occurred
PWM_CMP1	SM1STS[CMPF]	SM1INTEN[CMPIE]	Submodule 1 compare interrupt	Compare event has occurred
PWM_CAP1	SM1STS[CFA1], SM1STS[CFA0], SM1STS[CFB1], SM1STS[CFB0], SM1STS[CFX1], SM1STS[CFX0]	SM1INTEN[CFA1IE], SM1INTEN[CFA0IE], SM1INTEN[CFB1IE], SM1INTEN[CFB0IE], SM1INTEN[CFX1IE], SM1INTEN[CFX0IE]	Submodule 1 input capture interrupt	Input capture event has occurred
PWM_RELOAD1	SM1STS[RF]	SM1INTEN[RIE]	Submodule 1 reload interrupt	Reload event has occurred
PWM_CMP2	SM2STS[CMPF]	SM2INTEN[CMPIE]	Submodule 2 compare interrupt	Compare event has occurred
PWM_CAP2	SM2STS[CFA1], SM2STS[CFA0], SM2STS[CFB1], SM2STS[CFB0], SM2STS[CFX1], SM2STS[CFX0]	SM2INTEN[CFA1IE], SM2INTEN[CFA0IE], SM2INTEN[CFB1IE], SM2INTEN[CFB0IE], SM2INTEN[CFX1IE], SM2INTEN[CFX0IE	Submodule 2 input capture interrupt	Input capture event has occurred
PWM_RELOAD2	SM2STS[RF]	SM2INTEN[RIE]	Submodule 2 reload interrupt	Reload event has occurred
PWM_CMP3	SM3STS[CMPF]	SM3INTEN[CMPIE]	Submodule 3 compare interrupt	Compare event has occurred

Table continues on the next page...

Table 28-3. Interrupt Summary (continued)

Core Interrupt	Interrupt Flag	Interrupt Enable	Name	Description
PWM_CAP3	SM3STS[CFA1], SM3STS[CFA0], SM3STS[CFB1], SM3STS[CFB0], SM3STS[CFX1], SM3STS[CFX0]	SM3INTEN[CFA1IE], SM3INTEN[CFA0IE], SM3INTEN[CFB1IE], SM3INTEN[CFB0IE], SM3INTEN[CFX1IE], SM3INTEN[CFX0IE	Submodule 3 input capture interrupt	Input capture event has occurred
PWM_RELOAD3	SM3STS[RF]	SM3INTEN[RIE]	Submodule 3 reload interrupt	Reload event has occurred
PWM_RERR	SM0STS[REF]	SM0INTEN[REIE]	Submodule 0 reload error interrupt	Reload error has occurred
	SM1STS[REF]	SM1INTEN[REIE]	Submodule 1 reload error interrupt	
	SM2STS[REF]	SM2INTEN[REIE]	Submodule 2 reload error interrupt	
	SM3STS[REF]	SM3INTEN[REIE]	Submodule 3 reload error interrupt	
PWM_FAULT	FSTS0[FFLAG], FSTS1[FFLAG]	FCTRL0[FIE], FCTRL1[FIE]	Fault input interrupt	Fault condition has been detected

28.7 DMA

Each submodule can request a DMA read access for its capture FIFOs and a DMA write request for its double buffered VALx registers.

Table 28-4. DMA Summary

DMA Request	DMA Enable	Name	Description
Submodule 0 read request	SM0DMAEN[CX0DE]	SM0 Capture FIFO X0 read request	SM0CVAL0 contains a value to be read
	SM0DMAEN[CX1DE]	SM0 Capture FIFO X1 read request	SM0CVAL1 contains a value to be read
	SM0DMAEN[CA0DE]	SM0 Capture FIFO A0 read request	SM0CVAL2 contains a value to be read
	SM0DMAEN[CA1DE]	SM0 Capture FIFO A1 read request	SM0CVAL3 contains a value to be read
	SM0DMAEN[CB0DE]	SM0 Capture FIFO B0 read request	SM0CVAL4 contains a value to be read
	SM0DMAEN[CB1DE]	SM0 Capture FIFO B1 read request	SM0CVAL5 contains a value to be read
	SM0DMAEN[CAPTDE]	SM0 Capture FIFO read request source select	Selects source of submodule0 read DMA request

Table continues on the next page...

Table 28-4. DMA Summary (continued)

DMA Request	DMA Enable	Name	Description
Submodule 0 write request	SM0DMAEN[VALDE]	SM0VALx write request	SM0VALx registers need to be updated
Submodule 1 read request	SM1DMAEN[CX0DE]	SM1 Capture FIFO X0 read request	SM1CVAL0 contains a value to be read
	SM1DMAEN[CX1DE]	SM1 Capture FIFO X! read request	SM1CVAL1 contains a value to be read
	SM1DMAEN[CA0DE]	SM1Capture FIFO A0 read request	SM1CVAL2 contains a value to be read
	SM1DMAEN[CA1DE]	SM1 Capture FIFO A1 read request	SM1CVAL3 contains a value to be read
	SM1DMAEN[CB0DE]	SM1 Capture FIFO B0 read request	SM1CVAL4 contains a value to be read
	SM1DMAEN[CB1DE]	SM1 Capture FIFO B1 read request	SM!CVAL5 contains a value to be read
	SM1DMAEN[CAPTDE]	SM1 Capture FIFO read request source select	Selects source of submodule1 read DMA request
Submodule 1 write request	SM1DMAEN[VALDE]	SM1VALx write request	SM1VALx registers need to be updated
Submodule 2 read request	SM2DMAEN[CX0DE]	SM2 Capture FIFO X0 read request	SM2CVAL0 contains a value to be read
	SM2DMAEN[CX1DE]	SM2 Capture FIFO X! read request	SM2CVAL1 contains a value to be read
	SM2DMAEN[CA0DE]	SM2 Capture FIFO A0 read request	SM2CVAL2 contains a value to be read
	SM2DMAEN[CA1DE]	SM2 Capture FIFO A1 read request	SM2CVAL3 contains a value to be read
	SM2DMAEN[CB0DE]	SM2 Capture FIFO B0 read request	SM2CVAL4 contains a value to be read
	SM2DMAEN[CB1DE]	SM2 Capture FIFO B1 read request	SM2CVAL5 contains a value to be read
	SM2DMAEN[CAPTDE]	SM2 Capture FIFO read request source select	Selects source of submodule2 read DMA request
Submodule 2 write request	SM2DMAEN[VALDE]	SM2VALx write request	SM2VALx registers need to be updated
Submodule 3 read request	SM3DMAEN[CX0DE]	SM3 Capture FIFO X0 read request	SM3CVAL0 contains a value to be read
	SM3DMAEN[CX1DE]	SM3 Capture FIFO X! read request	SM3CVAL1 contains a value to be read
	SM3DMAEN[CA0DE]	SM3 Capture FIFO A0 read request	SM3CVAL2 contains a value to be read
	SM3DMAEN[CA1DE]	SM3 Capture FIFO A1 read request	SM3CVAL3 contains a value to be read
	SM3DMAEN[CB0DE]	SM3 Capture FIFO B0 read request	SM3CVAL4 contains a value to be read

Table continues on the next page...

Table 28-4. DMA Summary (continued)

DMA Request	DMA Enable	Name	Description
	SM3DMAEN[CB1DE]	SM3 Capture FIFO B1 read request	SM3CVAL5 contains a value to be read
	SM3DMAEN[CAPTDE]	SM3 Capture FIFO read request source select	Selects source of submodule3 read DMA request
Submodule 3 write request	SM3DMAEN[VALDE]	SM3VALx write request	SM3VALx registers need to be updated

Chapter 29
Programmable Delay Block (PDB)

29.1 Introduction

Inverter control and wireless charging applications often need to synchronize the time at which ADC samples or Comparator sampling windows are taken with respect to the PWM period. Normally the PWM timer module has a SYNC output specifically for that purpose. The primary function of the programmable delay block is simply to provide a controllable delay from the PWM SYNC output to the sample trigger input of the programmable gain amplifiers and ADCs as well as a controllable window that is synchronized with PWM pulses for analog comparators to compare the analog signals in a defined window. Another primary function of the PDB is to generate a sampling/filter clock that can be used by the comparator.

An alternate function of the PDB is to generate a PWM pulse that is synchronized to PWM timer module.

29.1.1 Features

PDB features include:

- 16-bit resolution with prescaler
- Positive transition of trigger_in will initiate the counter
- Supports two trigger_out signals. Each has an independently controlled delay from sync_in
- Trigger outputs can be ORed together to schedule two conversions from one input trigger event

- Trigger outputs can be used to schedule precise edge placement for a pulsed output. This feature is used to generate the control signal for the HSCMP windowing feature (see description of High Speed Comparator module) and output to a package pin if needed
- Continuous trigger or single shot mode supported
- Bypass mode supported
- Each trigger output is independently enabled

29.1.2 Modes of Operation

Modes of operation include:

- Disabled: Counter is off and TriggerA–D are low.
- Enabled OneShot: Counter is enabled and restarted at count one upon receiving a positive edge on the trigger input. TriggerA–D will see only one output trigger per input trigger.
- Enabled Continuous: Counter is enabled and restarted at count one. The counter will be rolled over to one again when the count reaches the value specified in the MOD register, and counting restarted. This enables a continuous stream of triggers out as a result of a single trigger input.
- Bypassed: The input trigger bypasses the PDB logic entirely. It is possible to bypass any of the trigger outputs or all.
- In Enabled OneShot and Enabled Continuous, the outputs of the DelayA and DelayB comparators and the DelayC and DelayD comparators can be combined in such a way that two ADC events can be triggered from a single input event. These will be referred to as TwoShot and Continuous TwoShot modes.
- In Enabled OneShot and Enabled Continuous, the outputs of the DelayA and DelayB comparators and the DelayC and DelayD comparators can be combined in such a way that an output pulse(s) can be generated with precisely controlled rising and falling times. These will be referred to as Single Pulse and Continuous Pulse modes.

29.1.3 Block Diagram

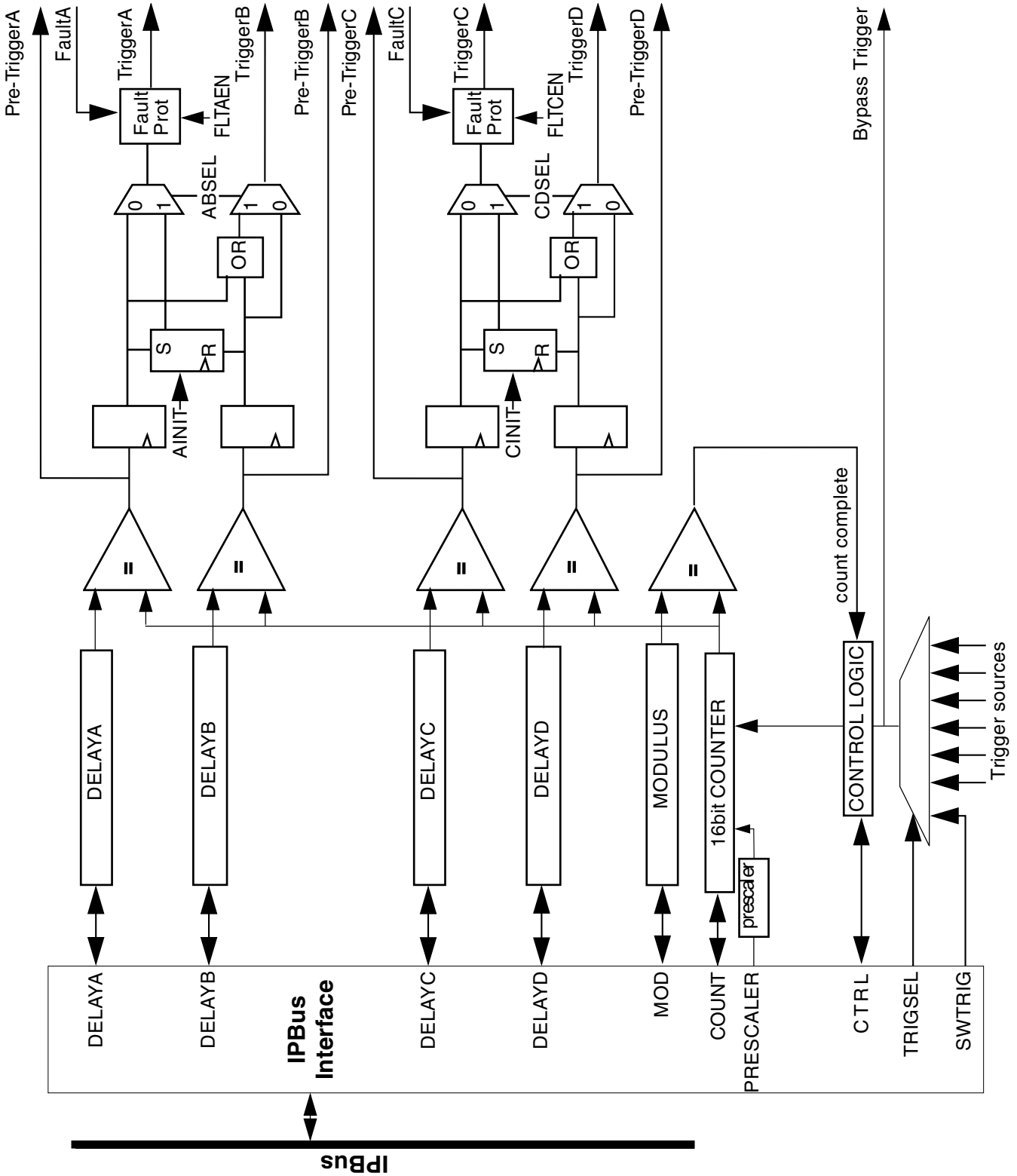


Figure 29-1. PDB Block Diagram

The preceding diagram shows the basic structure of the PDB block. The logic used to generate TriggerA and TriggerB is repeated for TriggerC and TriggerD. There is a single counter whose output is compared against the different delay values and the single modulus value.

DELAYA and DELAYB determine the time between assertion of the trigger input to the point at which changes in the trigger output signals are initiated. These times are defined as:

- trigger input to Pre-TriggerA = (prescaler X DELAYA) + 1 peripheral bus clock cycles
- trigger input to Pre-TriggerB = (prescaler X DELAYB) + 1 peripheral bus clock cycles
- Add one additional peripheral bus clock cycle to determine the time at which the trigger outputs change.

If the ADC block contains duplicate control and result registers, TriggerA and TriggerB allow them to operate in a ping-pong fashion, alternating conversions between two different analog sources (per converter). The Pre-Trigger signals are used to specify which signal is sampled next. Pre-TriggerA and Pre-TriggerB are used to precondition the PGA/ADC blocks one peripheral bus clock period prior to the actual measurement trigger.

The following figure shows the signals used to operate the ADC to sample signal A and sample single B in One-Shot mode. The trigger delays for the ADC are independently set via the DELAYA and DELAYB values. The one trigger signal from the PWM timer module can start the ADC twice with two different delays. However, the time between TriggerA and TriggerB must be longer than the ADC conversion time; then the second trigger signal can take effect.

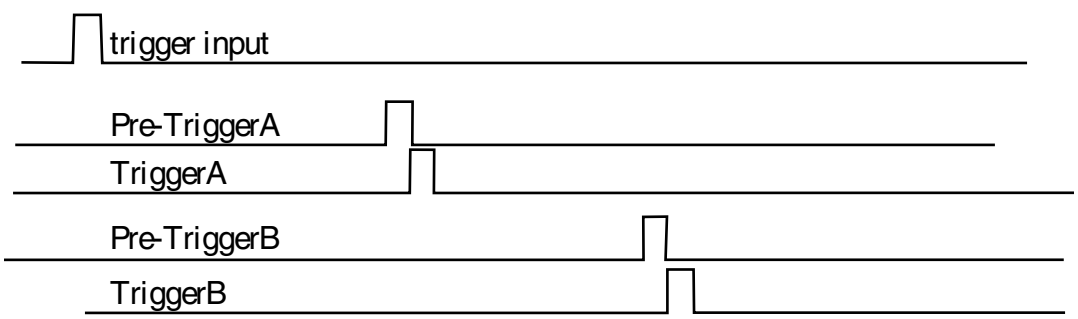


Figure 29-2. Decoupled A and B Trigger Generation

The following figure shows Two-Shot mode, when the device integrates two ADCs. In this case, both ADC A and ADC B are given the same trigger, resulting in a total of four ADC conversions (two on ADC A, and two on ADC B).

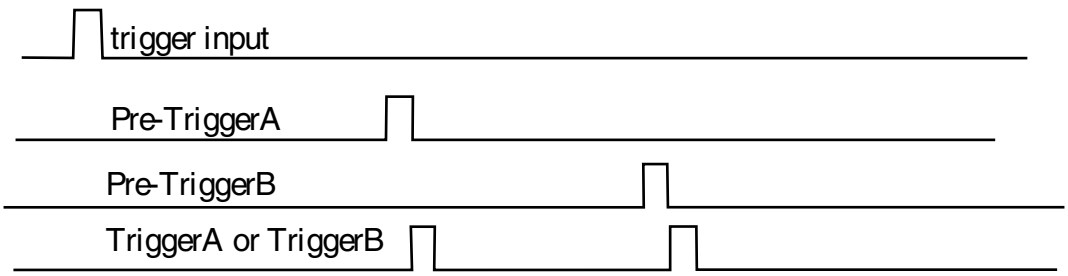


Figure 29-3. Trigger Configured for Simultaneous Sampling / Ping-Pong

The modulus value, MOD, is used to reset the counter back to 0x0001 at the end of the count. If MCTRL[CONT] is set, the counter will then resume a new count. Otherwise, the timer operation will cease until the next trigger input event occurs.

The following figure shows the pulsed modes. In this case, Pre-TriggerA and Pre-TriggerB are used to precisely schedule the rising and falling edges for the output waveform.

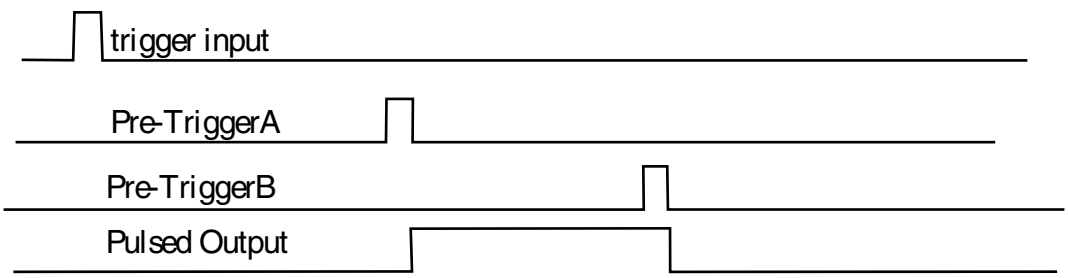


Figure 29-4. Trigger Pulsed Output Operation

The DELAY and MOD registers are buffered. The values written into these registers will not take effect until:

- A logic 1 is written to MCTRL[LDOK] if MCTRL[LDMOD] = 0.
- Either counter rolls over in continuous mode or trigger signal is received in one shot mode after a logic 1 is written to MCTRL[LDOK] if MCTRL[LDMOD] = 1. In this case, if any value is written to any one of these registers, after a logic 1 being written to MCTRL[LDOK], it will be ignored until the values in these registers are loaded into the buffers.

MCTRL[LDOK] will remain logic 1 after being set until the values in the DELAY and MOD registers are loaded into buffers. This bit is readable.

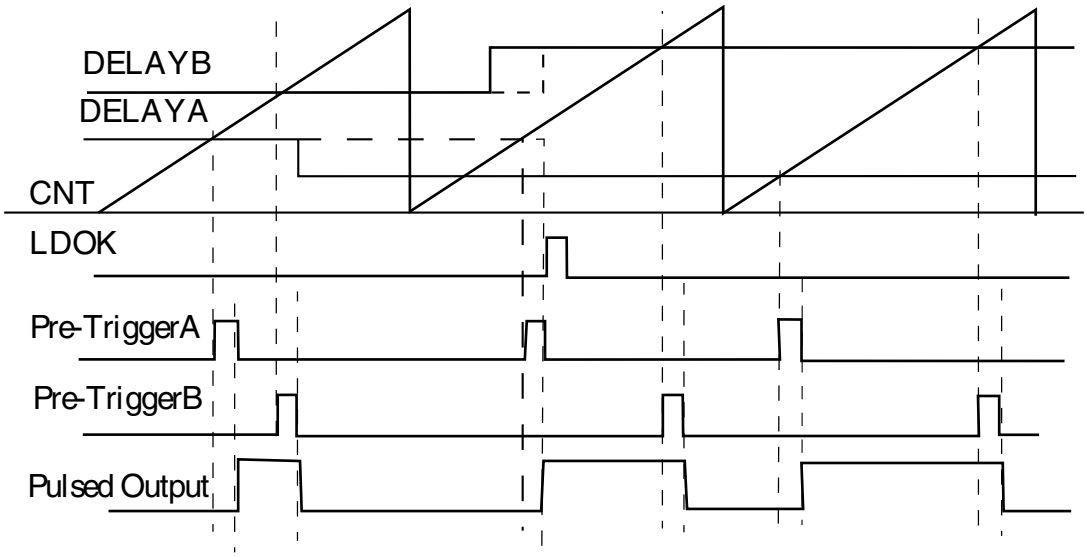


Figure 29-5. Registers Update with LDMOD bit = 0

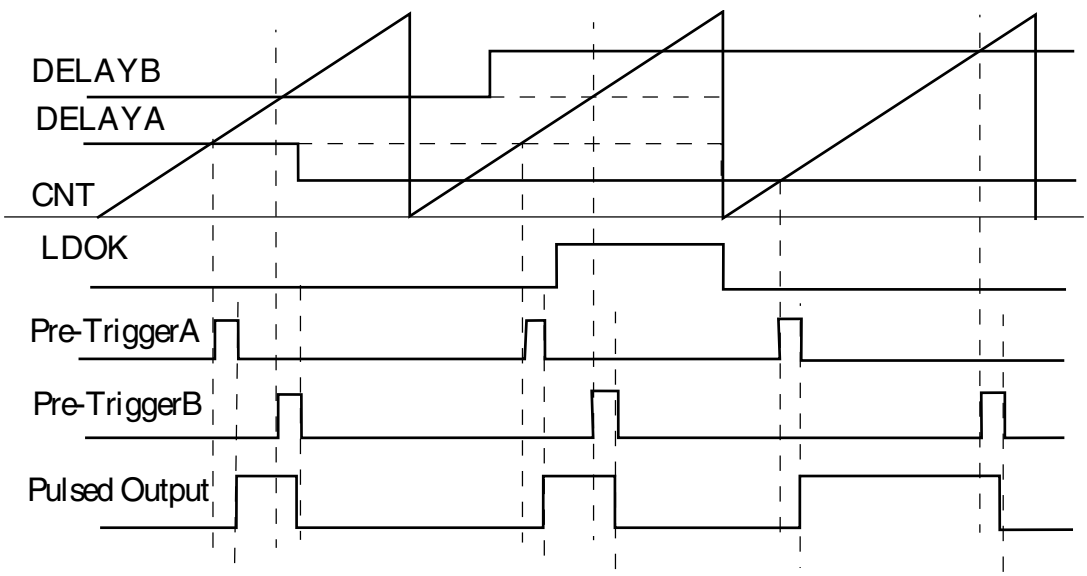


Figure 29-6. Registers Update with LDMOD bit = 1

29.2 Memory Map and Registers

PDB memory map

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
E120	Master Control Register (PDB0_MCTRL)	16	R/W	0000h	29.2.1/725
E121	Control A Register (PDB0_CTRLA)	16	R/W	0000h	29.2.2/727
E122	Control C Register (PDB0_CTRLC)	16	R/W	0000h	29.2.3/729

Table continues on the next page...

PDB memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
E123	DelayA Register (PDB0_DELAYA)	16	R/W	0000h	29.2.4/730
E124	DelayB Register (PDB0_DELAYB)	16	R/W	0000h	29.2.5/731
E125	DelayC Register (PDB0_DELAYC)	16	R/W	0000h	29.2.6/732
E126	DelayD Register (PDB0_DELAYD)	16	R/W	0000h	29.2.7/732
E127	Modulus Register (PDB0_MOD)	16	R/W	FFFFh	29.2.8/733
E128	Counter Register (PDB0_CNTR)	16	R	0001h	29.2.9/734
E130	Master Control Register (PDB1_MCTRL)	16	R/W	0000h	29.2.1/725
E131	Control A Register (PDB1_CTRLA)	16	R/W	0000h	29.2.2/727
E132	Control C Register (PDB1_CTRLC)	16	R/W	0000h	29.2.3/729
E133	DelayA Register (PDB1_DELAYA)	16	R/W	0000h	29.2.4/730
E134	DelayB Register (PDB1_DELAYB)	16	R/W	0000h	29.2.5/731
E135	DelayC Register (PDB1_DELAYC)	16	R/W	0000h	29.2.6/732
E136	DelayD Register (PDB1_DELAYD)	16	R/W	0000h	29.2.7/732
E137	Modulus Register (PDB1_MOD)	16	R/W	FFFFh	29.2.8/733
E138	Counter Register (PDB1_CNTR)	16	R	0001h	29.2.9/734

29.2.1 Master Control Register (PDBx_MCTRL)

This register contains control bits for the Programmable Delay Block.

Address: Base address + 0h offset

Bit	15	14	13	12	11	10	9	8
Read Write	PRESCALER			LDMOD	LDOK	CONT	COF	COIE
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
Read	0	TRIGSEL			0			PDBEN
Write	SWTRIG							
Reset	0	0	0	0	0	0	0	0

PDBx_MCTRL field descriptions

Field	Description
15–13 PRESCALER	Clock Prescaler Select 000 Timer uses peripheral clock. 001 Timer uses peripheral clock / 2. 010 Timer uses peripheral clock / 4. 011 Timer uses peripheral clock / 8. 100 Timer uses peripheral clock / 16.

Table continues on the next page...

PDBx_MCTRL field descriptions (continued)

Field	Description
	101 Timer uses peripheral clock / 32. 110 Timer uses peripheral clock / 64. 111 Timer uses peripheral clock / 128.
12 LDMOD	Load Mode Select 0 DELAY* and MOD registers are loaded into a set of buffers and take effect immediately after logic 1 is written to the MCTRL[LDOK] bit. 1 DELAY* and MOD registers are loaded into a set of buffers and take effect when the counter rolls over or a trigger signal is received after logic 1 is written to the MCTRL[LDOK] bit.
11 LDOK	Load OK Writing logic 1 to this bit loads the DELAY* and MOD registers into a set of buffers. • When MCTRL[LDMOD] = 0, the buffered delay and modulus values take effect immediately. • When MCTRL[LDMOD] = 1, the buffered delay and modulus values take effect when the counter rolls over in continuous mode or when a trigger signal is received in one-shot mode. After a logic 1 is written to MCTRL[LDOK], any value written to the DELAY* or MOD registers is ignored until the values in these registers are loaded into the buffers. This bit is cleared when buffers are loaded. Writing logic 0 to this bit has no effect. Reading this bit can determine if the values in the DELAY* and MOD registers have been loaded to the buffers and taken effect.
10 CONT	Continuous Mode Enable 0 Module is in one-shot mode. 1 Module is in continuous mode.
9 COF	Counter Overflow Flag This bit is set when a successful compare of the values of counter and modulus occurs and then the counter is rolled over. Clear this bit by writing logic one to it.
8 COIE	Counter Overflow Interrupt Enable 0 Counter roll over interrupt requests disabled. 1 Counter roll over interrupt requests enabled.
7 SWTRIG	Software Trigger When CTRL[TRIGSEL]=111 and the module is enabled, writing a one to this field triggers a reset and restarts the counter. Alternatively, if TriggerA or TriggerB is bypassed, it will propagate there immediately. This bit always reads as zero and, if passed to output triggers via the bypass mode, will have a one cycle pulse width.
6–4 TRIGSEL	Input Trigger Select 000 TriggerIn0 is selected. 001 TriggerIn1 is selected. 010 TriggerIn2 is selected. 011 TriggerIn3 is selected. 100 TriggerIn4 is selected. 101 TriggerIn5 is selected. 110 TriggerIn6 is selected. 111 SWTRIG is selected.

Table continues on the next page...

PDBx_MCTRL field descriptions (continued)

Field	Description
3–1 Reserved	Reserved This field is reserved. This read-only field is reserved and always has the value 0.
0 PDBEN	PDB Module Enable 0 Counter is off and all Trigger and PreTrigger outputs are low. 1 Counter is enabled.

29.2.2 Control A Register (PDBx_CTRLA)

This register contains control bits for the Trigger A and Trigger B outputs.

Address: Base address + 1h offset

Bit	15	14	13	12	11	10	9	8
Read	FLTA	FLTAEN	FPOLA	FLENA	DAF	DAIE	DBF	DBIE
Write								
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
Read	0		AINIT	ABSEL	BYPA	BYPB	ENA	ENB
Write								
Reset	0	0	0	0	0	0	0	0

PDBx_CTRLA field descriptions

Field	Description
15 FLTA	Fault A Input Status This read-only bit indicates the state of the Fault A input.
14 FLTAEN	Fault A Enable 0 Fault A input is ignored. 1 A logic 1 on the Fault A input forces TriggerA output to CTRLA[AINIT] until a counter reload occurs.
13 FPOLA	Fault A Polarity 0 A logic 0 on Fault A indicates a fault condition 1 A logic 1 on Fault A indicates a fault condition.
12 FLENA	Fault A Length This bit is used to determine the minimum width requirement of the input fault for it to be recognized as a valid fault condition. 0 Fault input must be active at least 2 IPBus clock cycles. 1 Fault input must be active at least 4 IPBus clock cycles.

Table continues on the next page...

PDBx_CTRLA field descriptions (continued)

Field	Description
11 DAF	Delay A Flag This bit is set when a successful compare of the values of counter and DELAYA occurs. Clear this bit by writing logic one to it.
10 DAIE	Delay A Interrupt Enable 0 DELAYA successful compare interrupt requests disabled. 1 DELAYA successful compare interrupt requests enabled.
9 DBF	Delay B Flag This bit is set when a successful compare of the values of counter and DELAYB occurs. Clear this bit by writing logic one to it.
8 DBIE	Delay B Interrupt Enable 0 DELAYB successful compare interrupt requests disabled. 1 DELAYB successful compare interrupt requests enabled.
7–6 Reserved	Reserved This field is reserved. This read-only field is reserved and always has the value 0.
5 AINIT	Initial Value A This bit is the value the Trigger A output is forced to when Fault A is active and CTRLA[FLTAEN] is set. When CTRLA[ABSEL] is set, this bit value is also forced onto the Trigger A output whenever the counter is reloaded.
4 ABSEL	Trigger A Output Select 0 Trigger A is a function of DELAYA only. Trigger B is a function of DELAYB only. 1 Trigger A and Trigger B outputs are a function of combined DELAYA and DELAYB. Trigger A is an extended pulse (as in trigger pulsed output operation) and Trigger B is a dual pulse (as in two-shot mode with trigger configured for simultaneous sampling).
3 BYPA	Bypass A 0 Trigger A is generated normally. 1 Trigger A generation is bypassed and Trigger A is a single pulse created by the selected trigger source.
2 BYPB	Bypass B 0 Trigger B is generated normally. 1 Trigger B generation is bypassed and Trigger B is a single pulse created by the selected trigger source.
1 ENA	Trigger A Enable 0 Trigger A is disabled and forced to 0. 1 Trigger A is enabled.
0 ENB	Trigger B Enable 0 Trigger B is disabled and forced to 0. 1 Trigger B is enabled.

29.2.3 Control C Register (PDBx_CTRLC)

This register contains control bits for the TriggerC and TriggerD outputs.

Address: Base address + 2h offset

Bit	15	14	13	12	11	10	9	8
Read	FLTC	FLTCEN	FPOLC	FLENC	DCF	DCIE	DDF	DDIE
Write								
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
Read	0		CINIT	CDSEL	BYPC	BYPD	ENC	END
Write								
Reset	0	0	0	0	0	0	0	0

PDBx_CTRLC field descriptions

Field	Description
15 FLTC	Fault C Input Status This read-only bit indicates the state of the Fault C input.
14 FLTCEN	Fault C Enable. 0 Fault C input is ignored. 1 A logic 1 on the Fault C input forces Trigger C output to CTRLC[CINIT] until a counter reload occurs.
13 FPOLC	Fault C Polarity 0 A logic 0 on Fault C indicates a fault condition. 1 A logic 1 on Fault C indicates a fault condition.
12 FLENC	Fault C Length This bit is used to determine the minimum width requirement of the input fault for it to be recognized as a valid fault condition. 0 Fault input must be active at least 2 IPBus clock cycles. 1 Fault input must be active at least 4 IPBus clock cycles.
11 DCF	Delay C Flag This bit is set when a successful compare of the values of counter and DELAYC occurs. Clear this bit by writing logic one to it.
10 DCIE	Delay C Interrupt Enable 0 DELAYC successful compare interrupt requests disabled. 1 DELAYC successful compare interrupt requests enabled.
9 DDF	Delay D Flag This bit is set when a successful compare of the values of counter and DELAYD occurs. Clear this bit by writing logic one to it.

Table continues on the next page...

PDBx_CTRLC field descriptions (continued)

Field	Description
8 DDIE	Delay D Interrupt Enable 0 DELAYD successful compare interrupt requests disabled. 1 DELAYD successful compare interrupt requests enabled.
7–6 Reserved	Reserved This field is reserved. This read-only field is reserved and always has the value 0.
5 CINIT	Initial Value C This bit is the value the Trigger C output is forced to when Fault C is active and CTRLC[FLTCEN] is set. When CTRLC[CDSEL] is set, this bit value is also forced onto the Trigger C output whenever the counter is reloaded.
4 CDSEL	Trigger C Output Select 0 Trigger C is a function of DELAYC only. Trigger D is a function of DELAYD only. 1 Trigger C and Trigger D outputs are a function of combined DELAYC and DELAYD. Trigger C is an extended pulse (as in trigger pulsed output operation) and Trigger D is a dual pulse (as in two-shot mode with trigger configured for simultaneous sampling).
3 BYPC	Bypass C 0 Trigger C is generated normally. 1 Trigger C generation is bypassed and Trigger C is a single pulse created by the selected trigger source.
2 BYPD	Bypass D 0 Trigger D is generated normally. 1 Trigger D generation is bypassed and Trigger D is a single pulse created by the selected trigger source.
1 ENC	Trigger C Enable 0 Trigger C is disabled and forced to 0. 1 Trigger C is enabled.
0 END	Trigger D Enable 0 Trigger D is disabled and forced to 0. 1 Trigger D is enabled.

29.2.4 DelayA Register (PDBx_DELAYA)

This register is used to specify the delay from assertion of TriggerIn to assertion of TriggerA out. This delay is only applicable if the module is enabled and the output trigger has been enabled and not been bypassed.

The DELAYA value is buffered. Writing to this register writes the data into a buffer, where it is held depending on the value of MCTRL[LDMOD].

- When MCTRL[LDMOD] = 0, the buffered value in this register takes effect immediately upon MCTRL[LDOK] being set.
- When MCTRL[LDMOD] = 1, the buffered value in this register is not used until the counter rolls over (in continuous mode) or until a trigger occurs (in triggered mode) and MCTRL[LDOK] = 1.

Restriction

Writes to the DELAYA register are ignored when MCTRL[LDOK] = 1.

Address: Base address + 3h offset

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read Write	DELAYA															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PDBx_DELAYA field descriptions

Field	Description
DELAYA	Delay A The delay is in terms of peripheral clock cycles. A delay value of $0000 will never be reached and no output trigger will occur since the counter goes from $0001 to $FFFF.

29.2.5 DelayB Register (PDBx_DELAYB)

This register is used to specify the delay from assertion of TriggerIn to assertion of TriggerB out. This delay is only applicable if the module is enabled and the output trigger has been enabled and not been bypassed.

The DELAYB value is buffered. Writing to this register writes the data into a buffer, where it is held depending on the value of MCTRL[LDMOD].

- When MCTRL[LDMOD] = 0, the buffered value in this register takes effect immediately upon MCTRL[LDOK] being set.
- When MCTRL[LDMOD] = 1, the buffered value in this register is not used until the counter rolls over (in continuous mode) or until a trigger occurs (in triggered mode) and MCTRL[LDOK] = 1.

Restriction

Writes to the DELAYB register are ignored when MCTRL[LDOK] = 1.

Address: Base address + 4h offset

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read Write	DELAYB															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PDBx_DELAYB field descriptions

Field	Description
DELAYB	Delay B The delay is in terms of peripheral clock cycles. A delay value of $0000 will never be reached and no output trigger will occur since the counter goes from $0001 to $FFFF.

29.2.6 DelayC Register (PDBx_DELAYC)

This register is used to specify the delay from assertion of TriggerIn to assertion of TriggerC out. This delay is only applicable if the module is enabled and the output trigger has been enabled and not been bypassed.

The DELAYC value is buffered. Writing to this register writes the data into a buffer, where it is held depending on the value of MCTRL[LDMOD].

- When MCTRL[LDMOD] = 0, the buffered value in this register takes effect immediately upon MCTRL[LDOK] being set.
- When MCTRL[LDMOD] = 1, the buffered value in this register is not used until the counter rolls over (in continuous mode) or until a trigger occurs (in triggered mode) and MCTRL[LDOK] = 1.

Restriction

Writes to the DELAYC register are ignored when MCTRL[LDOK] = 1.

Address: Base address + 5h offset

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read Write	DELAYC															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PDBx_DELAYC field descriptions

Field	Description
DELAYC	Delay C The delay is in terms of peripheral clock cycles. A delay value of $0000 will never be reached and no output trigger will occur since the counter goes from $0001 to $FFFF.

29.2.7 DelayD Register (PDBx_DELAYD)

This register is used to specify the delay from assertion of TriggerIn to assertion of TriggerD out. This delay is only applicable if the module is enabled and the output trigger has been enabled and not been bypassed.

The DELAYD value is buffered. Writing to this register writes the data into a buffer, where it is held depending on the value of MCTRL[LDMOD].

- When MCTRL[LDMOD] = 0, the buffered value in this register takes effect immediately upon MCTRL[LDOK] being set.
- When MCTRL[LDMOD] = 1, the buffered value in this register is not used until the counter rolls over (in continuous mode) or until a trigger occurs (in triggered mode) and MCTRL[LDOK] = 1.

Restriction

Writes to the DELAYD register are ignored when MCTRL[LDOK] = 1.

Address: Base address + 6h offset

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read Write	DELAYD															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PDBx_DELAYD field descriptions

Field	Description
DELAYD	Delay D The delay is in terms of peripheral clock cycles. A delay value of $0000 will never be reached and no output trigger will occur since the counter goes from $0001 to $FFFF.

29.2.8 Modulus Register (PDBx_MOD)

This register specifies the period of the counter in terms of peripheral bus cycles. When the counter reaches this value, it resets to 0x0001. If MCTRL[CONT] = 1, the count restarts.

The MOD value is buffered. Writing to this register writes the data into a buffer, where it is held depending on the value of MCTRL[LDMOD].

- When MCTRL[LDMOD] = 0, the buffered value takes effect immediately upon MCTRL[LDOK] being set.
- When MCTRL[LDMOD] = 1, the buffered value is not used until the counter rolls over (in continuous mode) or until a trigger occurs (in triggered mode) and MCTRL[LDOK] = 1.

Restriction

Writes to the MOD register are ignored when MCTRL[LDOK] = 1.

Address: Base address + 7h offset

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read Write	MOD															
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

PDBx_MOD field descriptions

Field	Description
MOD	Specifies the period of the counter in terms of peripheral bus cycles

29.2.9 Counter Register (PDBx_CNTR)

Address: Base address + 8h offset

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	COUNT															
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

PDBx_CNTR field descriptions

Field	Description
COUNT	This read-only register can be used to read the current value of the counter.

29.3 Functional Description

This section provides the PDB functional description.

29.3.1 Miscellaneous Concerns and SoC Integration

- The purposes of this block are to:
  - Manage the delay between an external event and the time at which comparator, ADC, or PGA sample(s) are taken
  - Generate a variable width pulse that is synchronized with PWM or timer modules
- Additional trigger events, after the first, will cause the counter to restart even if the counter is still counting from the previous trigger.

29.3.2 Impact of Using the Prescaler on Timing Resolution

Use of prescalers greater than 1 limit the count/delay accuracy in terms of peripheral clocks (to the modulus of the prescaler value). If the prescaler is set to div 2, then the only values of total peripheral clocks that can be detected are even values, if div is set to 4, then the only values of total peripheral clocks that can be decoded as detected are mod(4) and so forth. If the user wanted to set a really long delay value and used div 128, then he would be limited to a resolution of 128 bus clocks. Therefore, use the lowest possible prescaler for a given application.

29.3.3 Fault conditions

When a fault input is detected, the appropriate Trigger A or Trigger C output is forced to the value defined by CTRLA[AINIT] of CTRLC[CINIT]. The output remains in this state until the fault condition is removed and the counter is reloaded, either due to a modulus match and rollover or due to another input trigger.

Input faults must last at least 2 IPBus clock cycles in order to be recognized when CTRL*[FLEN*] is 0. They must last at least 4 IPBus clock cycles when CTRL*[FLEN*] is 1.

29.4 Resets

This module has a single reset input, corresponding to the chip-wide peripheral reset. After reset, all registers are set to their reset value.

29.5 Clocks

This module has a single clock input, the IP Bus peripheral clock.

29.6 Interrupts

This module has five interrupt sources

- DelayA successful compare

- DelayB successful compare
- DelayC successful compare
- DelayD successful compare
- Counter overflow

Each interrupt source has an enable bit that can block the interrupt request from getting recognized by interrupt controller. The module presents only one interrupt request to the interrupt controller. The interrupt request source should be determined by reading the corresponding interrupt flags in the control registers.

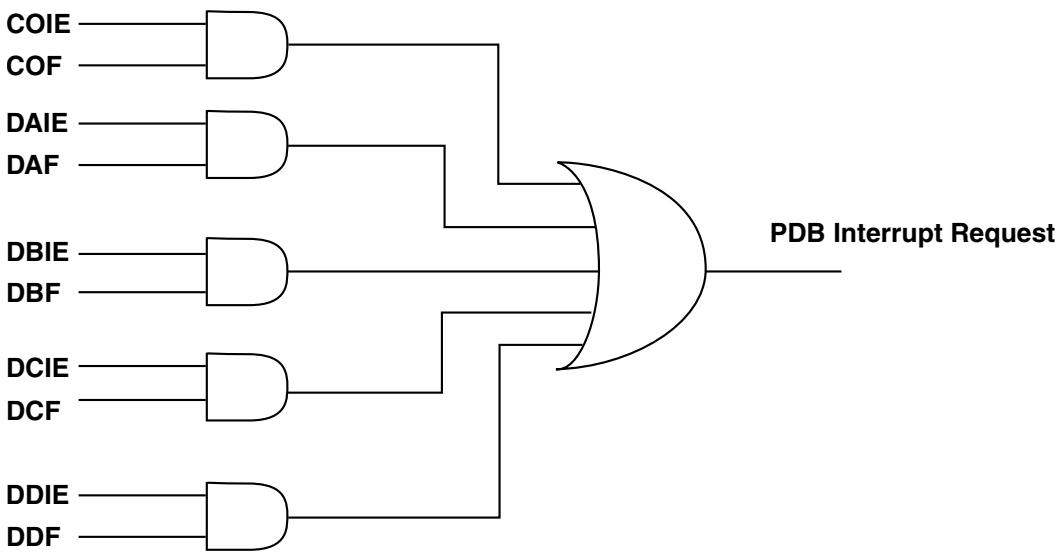


Figure 29-7. PDB Interrupt Request Generation

Chapter 30
Quad Timer (TMR)

30.1 Overview

Each timer module (TMR) contains four identical counter/timer groups. Each 16-bit counter/timer group contains a prescaler, a counter, a load register, a hold register, a capture register, two compare registers, two status and control registers, and one control register. All of the registers except the prescaler are read/writable.

NOTE

This document uses the terms "Timer" and "Counter" interchangeably because the counter/timers may perform either or both tasks.

The load register provides the initialization value to the counter when the counter's terminal value has been reached.

The hold register captures the counter's value when other counters are being read. This feature supports the reading of cascaded counters.

The capture register enables an external signal to take a "snap shot" of the counter's current value.

The COMP1 and COMP2 registers provide the values to which the counter is compared. If a match occurs, the OFLAG (TMR Output signal) can be set, cleared, or toggled. At match time, an interrupt is generated if enabled, and the new compare value is loaded into the COMP1 or COMP2 registers from CMPLD1 and CMPLD2 if enabled.

The prescaler provides different time bases useful for clocking the counter/timer.

The counter provides the ability to count internal or external events.

Within a timer module (set of four timer/counters), the input pins are shareable.

30.2 Features

The TMR module design includes these distinctive features:

- Four 16-bit counters/timers
- Count up/down
- Counters are cascadable
- Programmable count modulo
- Max count rate equals peripheral clock/2 for external clocks
- Max count rate equals peripheral clock for internal clocks
- Count once or repeatedly
- Counters are preloadable
- Compare registers are preloadable (available with compare load feature)
- Counters can share available input pins
- Separate prescaler for each counter
- Each counter has capture and compare capability
- Programmable operation during debug mode
- Inputs may act as fault inputs
- Programmable input filter
- Counting start can be synchronized across counters

30.3 Modes of Operation

The TMR module design operates in only a single mode of operation: Functional Mode. The various counting modes are detailed in the Functional Description.

30.4 Block Diagram

Each of the timer/counter groups within the quad-timer are shown in this figure.

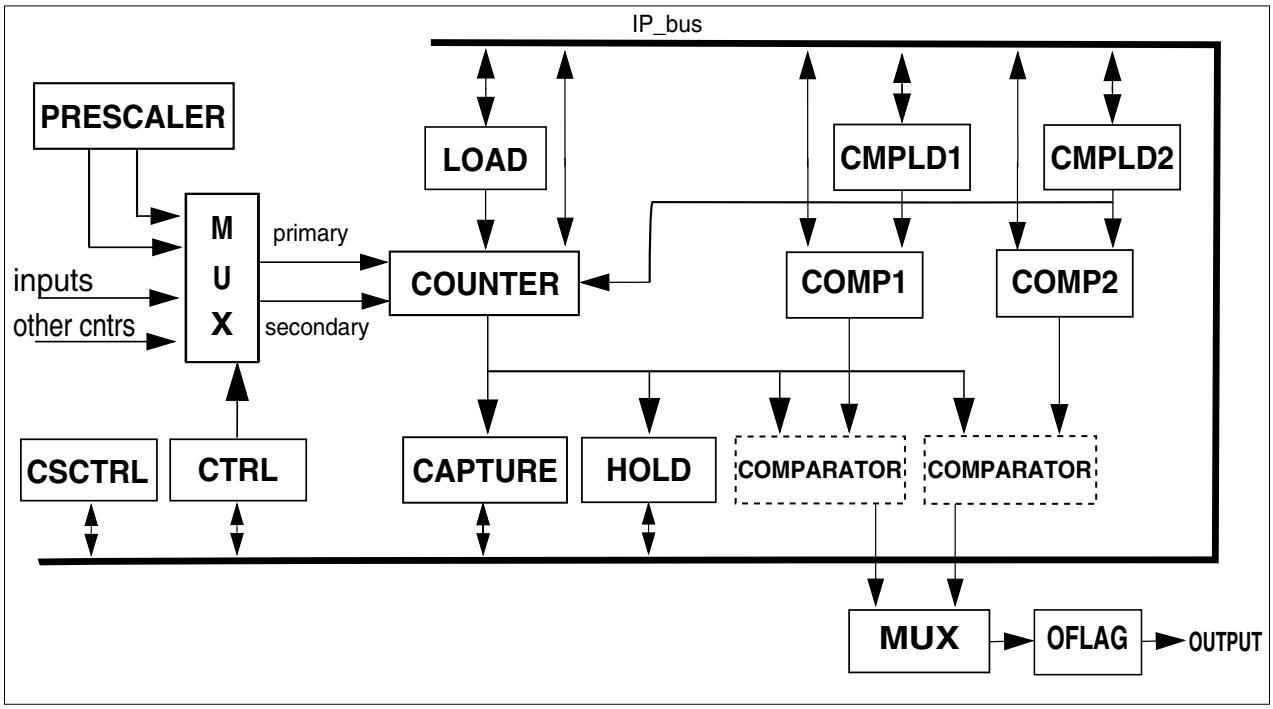


Figure 30-1. Quad Timer Block Diagram

30.5 Memory Map and Registers

The address of a register is the sum of a base address and an address offset. The base address is defined at the chip level and the address offset is defined at the module level. Make certain to check which quad timer is available on the chip being used, and which timer channels have external I/O.

TMR memory map

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
E140	Timer Channel Compare Register 1 (TMRA_0COMP1)	16	R/W	0000h	30.5.1/743
E141	Timer Channel Compare Register 2 (TMRA_0COMP2)	16	R/W	0000h	30.5.2/744
E142	Timer Channel Capture Register (TMRA_0CAPT)	16	R/W	0000h	30.5.3/744
E143	Timer Channel Load Register (TMRA_0LOAD)	16	R/W	0000h	30.5.4/744
E144	Timer Channel Hold Register (TMRA_0HOLD)	16	R/W	0000h	30.5.5/745
E145	Timer Channel Counter Register (TMRA_0CNTR)	16	R/W	0000h	30.5.6/745
E146	Timer Channel Control Register (TMRA_0CTRL)	16	R/W	0000h	30.5.7/745
E147	Timer Channel Status and Control Register (TMRA_0SCTRL)	16	R/W	0000h	30.5.8/748

Table continues on the next page...

TMR memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
E148	Timer Channel Comparator Load Register 1 (TMRA_0CMPLD1)	16	R/W	0000h	30.5.9/749
E149	Timer Channel Comparator Load Register 2 (TMRA_0CMPLD2)	16	R/W	0000h	30.5.10/ 750
E14A	Timer Channel Comparator Status and Control Register (TMRA_0CSCTRL)	16	R/W	0000h	30.5.11/ 750
E14B	Timer Channel Input Filter Register (TMRA_0FILT)	16	R/W	0000h	30.5.12/ 752
E14C	Timer Channel DMA Enable Register (TMRA_0DMA)	16	R/W	0000h	30.5.13/ 753
E14F	Timer Channel Enable Register (TMRA_0ENBL)	16	R/W	000Fh	30.5.14/ 754
E150	Timer Channel Compare Register 1 (TMRA_1COMP1)	16	R/W	0000h	30.5.1/743
E151	Timer Channel Compare Register 2 (TMRA_1COMP2)	16	R/W	0000h	30.5.2/744
E152	Timer Channel Capture Register (TMRA_1CAPT)	16	R/W	0000h	30.5.3/744
E153	Timer Channel Load Register (TMRA_1LOAD)	16	R/W	0000h	30.5.4/744
E154	Timer Channel Hold Register (TMRA_1HOLD)	16	R/W	0000h	30.5.5/745
E155	Timer Channel Counter Register (TMRA_1CNTR)	16	R/W	0000h	30.5.6/745
E156	Timer Channel Control Register (TMRA_1CTRL)	16	R/W	0000h	30.5.7/745
E157	Timer Channel Status and Control Register (TMRA_1SCTRL)	16	R/W	0000h	30.5.8/748
E158	Timer Channel Comparator Load Register 1 (TMRA_1CMPLD1)	16	R/W	0000h	30.5.9/749
E159	Timer Channel Comparator Load Register 2 (TMRA_1CMPLD2)	16	R/W	0000h	30.5.10/ 750
E15A	Timer Channel Comparator Status and Control Register (TMRA_1CSCTRL)	16	R/W	0000h	30.5.11/ 750
E15B	Timer Channel Input Filter Register (TMRA_1FILT)	16	R/W	0000h	30.5.12/ 752
E15C	Timer Channel DMA Enable Register (TMRA_1DMA)	16	R/W	0000h	30.5.13/ 753
E160	Timer Channel Compare Register 1 (TMRA_2COMP1)	16	R/W	0000h	30.5.1/743
E161	Timer Channel Compare Register 2 (TMRA_2COMP2)	16	R/W	0000h	30.5.2/744
E162	Timer Channel Capture Register (TMRA_2CAPT)	16	R/W	0000h	30.5.3/744
E163	Timer Channel Load Register (TMRA_2LOAD)	16	R/W	0000h	30.5.4/744
E164	Timer Channel Hold Register (TMRA_2HOLD)	16	R/W	0000h	30.5.5/745
E165	Timer Channel Counter Register (TMRA_2CNTR)	16	R/W	0000h	30.5.6/745
E166	Timer Channel Control Register (TMRA_2CTRL)	16	R/W	0000h	30.5.7/745
E167	Timer Channel Status and Control Register (TMRA_2SCTRL)	16	R/W	0000h	30.5.8/748
E168	Timer Channel Comparator Load Register 1 (TMRA_2CMPLD1)	16	R/W	0000h	30.5.9/749

Table continues on the next page...

TMR memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
E169	Timer Channel Comparator Load Register 2 (TMRA_2CMPLD2)	16	R/W	0000h	30.5.10/ 750
E16A	Timer Channel Comparator Status and Control Register (TMRA_2CSCTRL)	16	R/W	0000h	30.5.11/ 750
E16B	Timer Channel Input Filter Register (TMRA_2FILT)	16	R/W	0000h	30.5.12/ 752
E16C	Timer Channel DMA Enable Register (TMRA_2DMA)	16	R/W	0000h	30.5.13/ 753
E170	Timer Channel Compare Register 1 (TMRA_3COMP1)	16	R/W	0000h	30.5.1/743
E171	Timer Channel Compare Register 2 (TMRA_3COMP2)	16	R/W	0000h	30.5.2/744
E172	Timer Channel Capture Register (TMRA_3CAPT)	16	R/W	0000h	30.5.3/744
E173	Timer Channel Load Register (TMRA_3LOAD)	16	R/W	0000h	30.5.4/744
E174	Timer Channel Hold Register (TMRA_3HOLD)	16	R/W	0000h	30.5.5/745
E175	Timer Channel Counter Register (TMRA_3CNTR)	16	R/W	0000h	30.5.6/745
E176	Timer Channel Control Register (TMRA_3CTRL)	16	R/W	0000h	30.5.7/745
E177	Timer Channel Status and Control Register (TMRA_3SCTRL)	16	R/W	0000h	30.5.8/748
E178	Timer Channel Comparator Load Register 1 (TMRA_3CMPLD1)	16	R/W	0000h	30.5.9/749
E179	Timer Channel Comparator Load Register 2 (TMRA_3CMPLD2)	16	R/W	0000h	30.5.10/ 750
E17A	Timer Channel Comparator Status and Control Register (TMRA_3CSCTRL)	16	R/W	0000h	30.5.11/ 750
E17B	Timer Channel Input Filter Register (TMRA_3FILT)	16	R/W	0000h	30.5.12/ 752
E17C	Timer Channel DMA Enable Register (TMRA_3DMA)	16	R/W	0000h	30.5.13/ 753
E180	Timer Channel Compare Register 1 (TMRB_0COMP1)	16	R/W	0000h	30.5.1/743
E181	Timer Channel Compare Register 2 (TMRB_0COMP2)	16	R/W	0000h	30.5.2/744
E182	Timer Channel Capture Register (TMRB_0CAPT)	16	R/W	0000h	30.5.3/744
E183	Timer Channel Load Register (TMRB_0LOAD)	16	R/W	0000h	30.5.4/744
E184	Timer Channel Hold Register (TMRB_0HOLD)	16	R/W	0000h	30.5.5/745
E185	Timer Channel Counter Register (TMRB_0CNTR)	16	R/W	0000h	30.5.6/745
E186	Timer Channel Control Register (TMRB_0CTRL)	16	R/W	0000h	30.5.7/745
E187	Timer Channel Status and Control Register (TMRB_0SCTRL)	16	R/W	0000h	30.5.8/748
E188	Timer Channel Comparator Load Register 1 (TMRB_0CMPLD1)	16	R/W	0000h	30.5.9/749
E189	Timer Channel Comparator Load Register 2 (TMRB_0CMPLD2)	16	R/W	0000h	30.5.10/ 750
E18A	Timer Channel Comparator Status and Control Register (TMRB_0CSCTRL)	16	R/W	0000h	30.5.11/ 750

Table continues on the next page...

TMR memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
E18B	Timer Channel Input Filter Register (TMRB_0FILT)	16	R/W	0000h	30.5.12/ 752
E18C	Timer Channel DMA Enable Register (TMRB_0DMA)	16	R/W	0000h	30.5.13/ 753
E18F	Timer Channel Enable Register (TMRB_0ENBL)	16	R/W	000Fh	30.5.14/ 754
E190	Timer Channel Compare Register 1 (TMRB_1COMP1)	16	R/W	0000h	30.5.1/743
E191	Timer Channel Compare Register 2 (TMRB_1COMP2)	16	R/W	0000h	30.5.2/744
E192	Timer Channel Capture Register (TMRB_1CAPT)	16	R/W	0000h	30.5.3/744
E193	Timer Channel Load Register (TMRB_1LOAD)	16	R/W	0000h	30.5.4/744
E194	Timer Channel Hold Register (TMRB_1HOLD)	16	R/W	0000h	30.5.5/745
E195	Timer Channel Counter Register (TMRB_1CNTR)	16	R/W	0000h	30.5.6/745
E196	Timer Channel Control Register (TMRB_1CTRL)	16	R/W	0000h	30.5.7/745
E197	Timer Channel Status and Control Register (TMRB_1SCTRL)	16	R/W	0000h	30.5.8/748
E198	Timer Channel Comparator Load Register 1 (TMRB_1CMPLD1)	16	R/W	0000h	30.5.9/749
E199	Timer Channel Comparator Load Register 2 (TMRB_1CMPLD2)	16	R/W	0000h	30.5.10/ 750
E19A	Timer Channel Comparator Status and Control Register (TMRB_1CSCTRL)	16	R/W	0000h	30.5.11/ 750
E19B	Timer Channel Input Filter Register (TMRB_1FILT)	16	R/W	0000h	30.5.12/ 752
E19C	Timer Channel DMA Enable Register (TMRB_1DMA)	16	R/W	0000h	30.5.13/ 753
E1A0	Timer Channel Compare Register 1 (TMRB_2COMP1)	16	R/W	0000h	30.5.1/743
E1A1	Timer Channel Compare Register 2 (TMRB_2COMP2)	16	R/W	0000h	30.5.2/744
E1A2	Timer Channel Capture Register (TMRB_2CAPT)	16	R/W	0000h	30.5.3/744
E1A3	Timer Channel Load Register (TMRB_2LOAD)	16	R/W	0000h	30.5.4/744
E1A4	Timer Channel Hold Register (TMRB_2HOLD)	16	R/W	0000h	30.5.5/745
E1A5	Timer Channel Counter Register (TMRB_2CNTR)	16	R/W	0000h	30.5.6/745
E1A6	Timer Channel Control Register (TMRB_2CTRL)	16	R/W	0000h	30.5.7/745
E1A7	Timer Channel Status and Control Register (TMRB_2SCTRL)	16	R/W	0000h	30.5.8/748
E1A8	Timer Channel Comparator Load Register 1 (TMRB_2CMPLD1)	16	R/W	0000h	30.5.9/749
E1A9	Timer Channel Comparator Load Register 2 (TMRB_2CMPLD2)	16	R/W	0000h	30.5.10/ 750
E1AA	Timer Channel Comparator Status and Control Register (TMRB_2CSCTRL)	16	R/W	0000h	30.5.11/ 750
E1AB	Timer Channel Input Filter Register (TMRB_2FILT)	16	R/W	0000h	30.5.12/ 752

Table continues on the next page...

TMR memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
E1AC	Timer Channel DMA Enable Register (TMRB_2DMA)	16	R/W	0000h	30.5.13/ 753
E1B0	Timer Channel Compare Register 1 (TMRB_3COMP1)	16	R/W	0000h	30.5.1/743
E1B1	Timer Channel Compare Register 2 (TMRB_3COMP2)	16	R/W	0000h	30.5.2/744
E1B2	Timer Channel Capture Register (TMRB_3CAPT)	16	R/W	0000h	30.5.3/744
E1B3	Timer Channel Load Register (TMRB_3LOAD)	16	R/W	0000h	30.5.4/744
E1B4	Timer Channel Hold Register (TMRB_3HOLD)	16	R/W	0000h	30.5.5/745
E1B5	Timer Channel Counter Register (TMRB_3CNTR)	16	R/W	0000h	30.5.6/745
E1B6	Timer Channel Control Register (TMRB_3CTRL)	16	R/W	0000h	30.5.7/745
E1B7	Timer Channel Status and Control Register (TMRB_3SCTRL)	16	R/W	0000h	30.5.8/748
E1B8	Timer Channel Comparator Load Register 1 (TMRB_3CMPLD1)	16	R/W	0000h	30.5.9/749
E1B9	Timer Channel Comparator Load Register 2 (TMRB_3CMPLD2)	16	R/W	0000h	30.5.10/ 750
E1BA	Timer Channel Comparator Status and Control Register (TMRB_3CSCTRL)	16	R/W	0000h	30.5.11/ 750
E1BB	Timer Channel Input Filter Register (TMRB_3FILT)	16	R/W	0000h	30.5.12/ 752
E1BC	Timer Channel DMA Enable Register (TMRB_3DMA)	16	R/W	0000h	30.5.13/ 753

30.5.1 Timer Channel Compare Register 1 (TMR*x*_*n*COMP1)

Address: Base address + 0h offset + (16d × i), where i=0d to 3d

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read Write	COMPARISON_1															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

TMR*x*_*n*COMP1 field descriptions

Field	Description
COMPARISON_ 1	Comparison Value 1 This read/write register stores the value used for comparison with the counter value in count up mode.

30.5.2 Timer Channel Compare Register 2 (TMR*x*_*n*COMP2)

Address: Base address + 1h offset + (16d × i), where i=0d to 3d

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read Write	COMPARISON_2															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

TMR*x*_*n*COMP2 field descriptions

Field	Description
COMPARISON_2	Comparison Value 2 This read/write register stores the value used for comparison with the counter value in count down mode or alternating compare mode.

30.5.3 Timer Channel Capture Register (TMR*x*_*n*CAPT)

Address: Base address + 2h offset + (16d × i), where i=0d to 3d

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read Write	CAPTURE															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

TMR*x*_*n*CAPT field descriptions

Field	Description
CAPTURE	Capture Value This read/write register stores the value captured from the counter.

30.5.4 Timer Channel Load Register (TMR*x*_*n*LOAD)

Address: Base address + 3h offset + (16d × i), where i=0d to 3d

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read Write	LOAD															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

TMR*x*_*n*LOAD field descriptions

Field	Description
LOAD	Timer Load Register This read/write register stores the value used to initialize the counter after counter compare.

30.5.5 Timer Channel Hold Register (TMR*x*_*n*HOLD)

Address: Base address + 4h offset + (16d × i), where i=0d to 3d

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read Write	HOLD															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

TMR*x*_*n*HOLD field descriptions

Field	Description
HOLD	This read/write register stores the counter's values of specific channels whenever any of the four counters within a module is read.

30.5.6 Timer Channel Counter Register (TMR*x*_*n*CNTR)

Address: Base address + 5h offset + (16d × i), where i=0d to 3d

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read Write	COUNTER															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

TMR*x*_*n*CNTR field descriptions

Field	Description
COUNTER	This read/write register is the counter for the corresponding channel in a timer module.

30.5.7 Timer Channel Control Register (TMR*x*_*n*CTRL)

Address: Base address + 6h offset + (16d × i), where i=0d to 3d

Bit	15	14	13	12	11	10	9	8
Read Write	CM			PCS				SCS
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
Read Write	SCS	ONCE	LENGTH	DIR	COINIT	OUTMODE		
Reset	0	0	0	0	0	0	0	0

TMR*x*_*n*CTRL field descriptions

Field	Description
15–13 CM	Count Mode These bits control the basic counting and behavior of the counter.

Table continues on the next page...

TMRx_*n*CTRL field descriptions (continued)

Field	Description
	000 No operation 001 Count rising edges of primary source[1] 010 Count rising and falling edges of primary source[2] 011 Count rising edges of primary source while secondary input high active 100 Quadrature count mode, uses primary and secondary sources 101 Count rising edges of primary source; secondary source specifies direction[3] 110 Edge of secondary source triggers primary count until compare 111 Cascaded counter mode (up/down)[4]
12–9 PCS	Primary Count Source These bits select the primary count source. **NOTE:** A timer selecting its own output for input is not a legal choice. The result is no counting. 0000 Counter 0 input pin 0001 Counter 1 input pin 0010 Counter 2 input pin 0011 Counter 3 input pin 0100 Counter 0 output 0101 Counter 1 output 0110 Counter 2 output 0111 Counter 3 output 1000 IP bus clock divide by 1 prescaler 1001 IP bus clock divide by 2 prescaler 1010 IP bus clock divide by 4 prescaler 1011 IP bus clock divide by 8 prescaler 1100 IP bus clock divide by 16 prescaler 1101 IP bus clock divide by 32 prescaler 1110 IP bus clock divide by 64 prescaler 1111 IP bus clock divide by 128 prescaler
8–7 SCS	Secondary Count Source These bits identify the external input pin to be used as a count command or timer command. The selected input can trigger the timer to capture the current value of CNTR . The selected input can also be used to specify the count direction. The selected signal can also be used as a fault input when CSCTRL[FAULT] is set. The polarity of the signal can be inverted by SCTRL[IPS]. 00 Counter 0 input pin 01 Counter 1 input pin 10 Counter 2 input pin 11 Counter 3 input pin
6 ONCE	Count Once This bit selects continuous or one shot counting mode. 0 Count repeatedly. 1 Count until compare and then stop. If counting up, a successful compare occurs when the counter reaches a COMP1 value. If counting down, a successful compare occurs when the counter reaches a COMP2 value. When output mode $4 is used, the counter re-initializes after reaching the COMP1 value, continues to count to the COMP2 value, and then stops.

Table continues on the next page...

TMRx_*n*CTRL field descriptions (continued)

Field	Description
5 LENGTH	Count Length This bit determines whether the counter: • counts to the compare value and then re-initializes itself to the value specified in the LOAD (or CMPLD2) register, or • continues counting past the compare value to the binary roll over. 0 Count until roll over at $FFFF and continue from $0000. 1 Count until compare, then re-initialize. If counting up, a successful compare occurs when the counter reaches a COMP1 value. If counting down, a successful compare occurs when the counter reaches a COMP2 value. When output mode $4 is used, alternating values of COMP1 and COMP2 are used to generate successful comparisons. For example, the counter counts until a COMP1 value is reached, re-initializes, counts until COMP2 value is reached, re-initializes, counts until COMP1 value is reached, and so on.
4 DIR	Count Direction This bit selects either the normal count direction up, or the reverse direction, down. 0 Count up. 1 Count down.
3 COINIT	Co-Channel Initialization This bit enables another counter/timer within the module to force the re-initialization of this counter/timer when it has an active compare event. 0 Co-channel counter/timers cannot force a re-initialization of this counter/timer 1 Co-channel counter/timers may force a re-initialization of this counter/timer
OUTMODE	Output Mode These bits determine the mode of operation for the OFLAG output signal. 000 Asserted while counter is active 001 Clear OFLAG output on successful compare 010 Set OFLAG output on successful compare 011 Toggle OFLAG output on successful compare 100 Toggle OFLAG output using alternating compare registers 101 Set on compare, cleared on secondary source input edge 110 Set on compare, cleared on counter rollover 111 Enable gated clock output while counter is active

1. Rising edges are counted only when SCTRL[IPS] = 0. Falling edges are counted when SCTRL[IPS] = 1. If the primary count source is IP bus clock divide by 1, only rising edges are counted regardless of the value of SCTRL[IPS].
2. IP bus clock divide by 1 cannot be used as a primary count source in edge count mode.
3. Rising edges are counted only when SCTRL[IPS] = 0. Falling edges are counted when SCTRL[IPS] = 1.
4. The primary count source must be set to one of the counter outputs.

30.5.8 Timer Channel Status and Control Register (TMRx_*n*SCTRL)

Address: Base address + 7h offset + (16d × i), where i=0d to 3d

Bit	15	14	13	12	11	10	9	8
Read	TCF	TCFIE	TOF	TOFIE	IEF	IEFIE	IPS	INPUT
Write								
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
Read	CAPTURE_MODE		MSTR	EEOF	VAL	0	OPS	OEN
Write						FORCE		
Reset	0	0	0	0	0	0	0	0

TMRx_*n*SCTRL field descriptions

Field	Description
15 TCF	Timer Compare Flag This bit is set when a successful compare occurs. This bit is cleared by writing a zero to this bit location.
14 TCFIE	Timer Compare Flag Interrupt Enable This bit (when set) enables interrupts when TCF is set.
13 TOF	Timer Overflow Flag This bit is set when the counter rolls over its maximum value $FFFF or $0000 (depending on count direction). This bit is cleared by writing a zero to this bit location.
12 TOFIE	Timer Overflow Flag Interrupt Enable This bit (when set) enables interrupts when TOF is set.
11 IEF	Input Edge Flag This bit is set when CAPTMODE is enabled and a proper input transition occurs (on an input selected as a secondary count source) while the count mode does not equal 000. This bit is cleared by writing a zero to this bit position. This bit can also be cleared automatically by a read of CAPT when DMA[IEFDE] is set. **NOTE:** Setting the input polarity select bit (IPS) changes the edge to be detected. Also, the control register's secondary count source determines which external input pin is monitored by the detection circuitry.
10 IEFIE	Input Edge Flag Interrupt Enable This bit (when set) enables interrupts when IEF is set. **Restriction:** Do not set both this bit and DMA[IEFDE].
9 IPS	Input Polarity Select This bit (when set) inverts the input signal polarity.
8 INPUT	External Input Signal This read-only bit reflects the current state of the external input pin selected via the secondary count source after application of IPS and filtering.

Table continues on the next page...

TMRx_*n*SCTRL field descriptions (continued)

Field	Description
7–6 CAPTURE_ MODE	Input Capture Mode These bits specify the operation of the capture register as well as the operation of the input edge flag. The input source is the secondary count source. 00 Capture function is disabled 01 Load capture register on rising edge (when IPS=0) or falling edge (when IPS=1) of input 10 Load capture register on falling edge (when IPS=0) or rising edge (when IPS=1) of input 11 Load capture register on both edges of input
5 MSTR	Master Mode This bit (when set) enables the compare function's output to be broadcasted to the other counters/timers in the module. This signal then can be used to re-initialize the other counters and/or force their OFLAG signal outputs.
4 EEOF	Enable External OFLAG Force This bit (when set) enables the compare from another counter/timer within the same module to force the state of this counter's OFLAG output signal.
3 VAL	Forced OFLAG Value This bit determines the value of the OFLAG output signal when software triggers a FORCE command.
2 FORCE	Force OFLAG Output This write only bit forces the current value of VAL to be written to the OFLAG output. This bit always reads as a zero. VAL and FORCE can be written simultaneously in a single write operation. Write to FORCE only if the counter is disabled. Setting this bit while the counter is enabled may yield unpredictable results.
1 OPS	Output Polarity Select This bit determines the polarity of the OFLAG output signal. 0 True polarity. 1 Inverted polarity.
0 OEN	Output Enable This bit determines the direction of the external pin. 0 The external pin is configured as an input. 1 The OFLAG output signal is driven on the external pin. Other timer groups using this external pin as their input see the driven value. The polarity of the signal is determined by OPS.

30.5.9 Timer Channel Comparator Load Register 1 (TMRx_*n*CMPLD1)

Address: Base address + 8h offset + (16d × i), where i=0d to 3d

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read Write	COMPARATOR_LOAD_1															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

TMRx_nCMPLD1 field descriptions

Field	Description
COMPARATOR_LOAD_1	This read/write register is the comparator 1 preload value for the COMP1 register for the corresponding channel in a timer module.

30.5.10 Timer Channel Comparator Load Register 2 (TMRx_nCMPLD2)

Address: Base address + 9h offset + (16d × i), where i=0d to 3d

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read Write	COMPARATOR_LOAD_2															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

TMRx_nCMPLD2 field descriptions

Field	Description
COMPARATOR_LOAD_2	This read/write register is the comparator 2 preload value for the COMP2 register for the corresponding channel in a timer module.

30.5.11 Timer Channel Comparator Status and Control Register (TMRx_nCSCTRL)

Address: Base address + Ah offset + (16d × i), where i=0d to 3d

Bit	15	14	13	12	11	10	9	8
Read	DBG_EN		FAULT	ALT_LOAD	ROC	TCI	UP	0
Write								
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
Read	TCF2EN	TCF1EN	TCF2	TCF1	CL2		CL1	
Write								
Reset	0	0	0	0	0	0	0	0

TMRx_nCSCTRL field descriptions

Field	Description
15–14 DBG_EN	Debug Actions Enable These bits allow the TMR module to perform certain actions in response to the chip entering debug mode. 00 Continue with normal operation during debug mode. (default) 01 Halt TMR counter during debug mode.

Table continues on the next page...

TMRx_nCSCTRL field descriptions (continued)

Field	Description
	10 Force TMR output to logic 0 (prior to consideration of SCTRL[OPS]). 11 Both halt counter and force output to 0 during debug mode.
13 FAULT	Fault Enable The selected secondary input acts as a fault signal so that the timer OFLAG is cleared when the secondary input is set. When the secondary input is used in this mode, there is no resynchronization of the input so that there is a combinational path to clear the OFLAG. Fault inputs less than two clock periods wide will not be latched. Latched faults will be cleared the next time that the counter logic sets the OFLAG. 0 Fault function disabled. 1 Fault function enabled.
12 ALT_LOAD	Alternative Load Enable This bit allows for an alternative method for loading the counter during modulo counting. Normally, the counter can be loaded only with the value from the LOAD register. When this bit is set, the counter is loaded from the LOAD register when counting up and a match with COMP1 occurs, or the counter is loaded from the CMPLD2 register when counting down and a match with COMP2 occurs. 0 Counter can be re-initialized only with the LOAD register. 1 Counter can be re-initialized with the LOAD or CMPLD2 registers depending on count direction.
11 ROC	Reload on Capture This bit enables the capture function to cause the counter to be reloaded from the LOAD register. 0 Do not reload the counter on a capture event. 1 Reload the counter on a capture event.
10 TCI	Triggered Count Initialization Control This bit is used during triggered count mode, CTRL[CM] = 110, to enable the counter to be re-initialized when a second trigger occurs while the counter is still counting. Normally, the second trigger causes the counting to stop/pause until a third trigger occurs. With this bit set, a second trigger event causes the counter to re-initialize and continue counting. 0 Stop counter upon receiving a second trigger event while still counting from the first trigger event. 1 Reload the counter upon receiving a second trigger event while still counting from the first trigger event.
9 UP	Counting Direction Indicator This read-only bit is used during quadrature count mode, CTRL[CM] = 100, to read the direction of the last count. CTRL[DIR] reverses the sense of this bit. 0 The last count was in the DOWN direction. 1 The last count was in the UP direction.
8 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
7 TCF2EN	Timer Compare 2 Interrupt Enable An interrupt is issued when both this bit and TCF2 are set.
6 TCF1EN	Timer Compare 1 Interrupt Enable An interrupt is issued when both this bit and TCF1 are set.

Table continues on the next page...

TMRx_*n*CSCTRL field descriptions (continued)

Field	Description
5 TCF2	Timer Compare 2 Interrupt Flag When set, this bit indicates a successful comparison of the timer and the the COMP2 register has occurred. This bit is sticky, and will remain set until explicitly cleared by writing a zero to this bit location.
4 TCF1	Timer Compare 1 Interrupt Flag When set, this bit indicates a successful comparison of the timer and the the COMP1 register has occurred. This bit is sticky, and will remain set until explicitly cleared by writing a zero to this bit location.
3–2 CL2	Compare Load Control 2 These bits control when COMP2 is preloaded with the value from CMPLD2. 00 Never preload 01 Load upon successful compare with the value in COMP1 10 Load upon successful compare with the value in COMP2 11 Reserved
CL1	Compare Load Control 1 These bits control when COMP1 is preloaded with the value from CMPLD1. 00 Never preload 01 Load upon successful compare with the value in COMP1 10 Load upon successful compare with the value in COMP2 11 Reserved

30.5.12 Timer Channel Input Filter Register (TMRx_*n*FILT)

The FILT register programs the values for the filtering of the corresponding input without regard for the fact that any timer channel can use the input as a count source.

Input filter considerations:

- Set the FILT_PER value such that the sampling period is larger than the period of the expected noise. In this way, a noise spike will corrupt only one sample. Choose the FILT_CNT value to reduce the probability that noisy samples cause an incorrect transition to be recognized. The probability of an incorrect transistion is defined as the probability of an incorrect sample raised to the power of (FILT_CNT + 3).
- The values of FILT_PER and FILT_CNT must also be balanced against the desire for minimal latency in recognizing input transitions. Turning on the input filter (setting FILT_PER to a non-zero value) introduces a latency of (((FILT_CNT + 3) x FILT_PER) + 2) IP bus clock periods.

Address: Base address + Bh offset + (16d × i), where i=0d to 3d

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	0					FILT_CNT			FILT_PER							
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

TMRx_nFILT field descriptions

Field	Description
15–11 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
10–8 FILT_CNT	Input Filter Sample Count These bits represent the number of consecutive samples that must agree prior to the input filter accepting an input transition. A value of 0x0 represents 3 samples. A value of 0x7 represents 10 samples. The value of FILT_CNT affects the input latency.
FILT_PER	Input Filter Sample Period These bits represent the sampling period (in IP bus clock cycles) of the TMR input signals. Each input is sampled multiple times at the rate specified by this field. If FILT_PER is 0x00 (default), then the input filter is bypassed. The value of FILT_PER affects the input latency. When changing values for FILT_PER from one non-zero value to another non-zero value, write a value of zero first to clear the filter.

30.5.13 Timer Channel DMA Enable Register (TMRx_nDMA)

Address: Base address + Ch offset + (16d × i), where i=0d to 3d

Bit	15	14	13	12	11	10	9	8
Read	0							
Write								
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
Read	0					CMPLD2DE	CMPLD1DE	IEFDE
Write								
Reset	0	0	0	0	0	0	0	0

TMRx_nDMA field descriptions

Field	Description
15–3 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
2 CMPLD2DE	Comparator Preload Register 2 DMA Enable Setting this bit enables DMA write requests for CMPLD2 whenever data is transferred out of the CMPLD2 register into the CNTR or COMP2 registers.
1 CMPLD1DE	Comparator Preload Register 1 DMA Enable Setting this bit enables DMA write requests for CMPLD1 whenever data is transferred out of the CMPLD1 register into the COMP1 register.
0 IEFDE	Input Edge Flag DMA Enable Setting this bit enables DMA read requests for CAPT when SCTRL[IEF] is set. **Restriction:** Do not set both this bit and SCTRL[IEFIE].

30.5.14 Timer Channel Enable Register (TMR*x*_*n*ENBL)

Address: Base address + Fh offset + (0d × i), where i=0d to 0d

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	0												ENBL			
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1

TMR*x*_*n*ENBL field descriptions

Field	Description
15–4 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
ENBL	Timer Channel Enable These bits enable the prescaler (if it is being used) and counter in each channel. Multiple ENBL bits can be set at the same time to synchronize the start of separate counters. If an ENBL bit is set, then the corresponding channel starts its counter as soon as the CTRL[CM] field has a value other than 0. When an ENBL bit is clear, the corresponding counter maintains its current value. 0 Timer channel is disabled. 1 Timer channel is enabled. (default)

30.6 Functional Description

30.6.1 General

The counter/timer has two basic modes of operation: it can count internal or external events, or it can count an internal clock source while an external input signal is asserted, thus timing the width of the external input signal.

- The counter can count the rising, falling, or both edges of the selected input pin.
- The counter can decode and count quadrature encoded input signals.
- The counter can count up and down using dual inputs in a "count with direction" format.
- The counter's terminal count value (modulo) is programmable.
  - The value that is loaded into the counter after reaching its terminal count is programmable.

- The counter can count repeatedly, or it can stop after completing one count cycle.
- The counter can be programmed to count to a programmed value and then immediately reinitialize, or it can count through the compare value until the count "rolls over" to zero.

The external inputs to each counter/timer are shareable among each of the four counter/timers within the module. The external inputs can be used as:

- Count commands
- Timer commands
- They can trigger the current counter value to be "captured"
- They can be used to generate interrupt requests

The polarity of the external inputs are selectable.

The primary output of each timer/counter is the output signal OFLAG. The OFLAG output signal can be:

- Set, cleared, or toggled when the counter reaches the programmed value.
- The OFLAG output signal may be output to an external pin instead of having that pin serve as a timer input.
- The OFLAG output signal enables each counter to generate square waves, PWM, or pulse stream outputs.
- The polarity of the OFLAG output signal is selectable.

Any counter/timer can be assigned as a master. A master's compare signal can be broadcast to the other counter/timers within the module. The other counters can be configured to reinitialize their counters and/or force their OFLAG output signals to predetermined values when a master's counter/timer compare event occurs.

30.6.2 Usage of Compare Registers

The dual compare registers (COMP1 and COMP2) provide a bidirectional modulo count capability. The COMP1 register is used when the counter is *counting up*, and the COMP2 register is used when the counter is *counting down*. Alternating compare mode is the only exception.

The COMP1 register should be set to the desired maximum count value or FFFFh to indicate the maximum unsigned value prior to roll-over, and the COMP2 register should be set to the minimum count value or 0000h to indicate the minimum unsigned value prior to roll-under.

If CTRL[OUTMODE] is set to 100, the OFLAG will toggle while using alternating compare registers. In this variable frequency PWM mode, the COMP2 value defines the desired pulse width of the on time, and the COMP1 register defines the off time.

Use caution when changing COMP1 and COMP2 while the counter is active. If the counter has already passed the new value, it will count to FFFFh or 0000h, roll over, then begin counting toward the new value. The check is: Count=CMPx, *not* Count > COMP1 or Count < COMP2.

The use of the CMPLD1 and CMPLD2 registers to compare values will help to minimize this problem.

30.6.3 Usage of Compare Load Registers

The CMPLD1, CMPLD2, and CSCTRL registers offer a high degree of flexibility for loading compare registers with user-defined values on different compare events. To ensure correct functionality while using these registers we strongly suggest using the following method described in this section.

The purpose of the compare load feature is to allow quicker updating of the compare registers. In the past, a compare register could be updated using interrupts. However, because of the latency between an interrupt event occurring and the service of that interrupt, there was the possibility that the counter may have already counted past the new compare value by the time the compare register was updated by the interrupt service routine. The counter would then continue counting until it rolled over and reached the new compare value.

To address this, the compare registers are now updated in hardware in the same way the counter register is re-initialized to the value stored in the load register. The compare load feature allows the user to calculate new compare values and store them in to the comparator load registers. When a compare event occurs, the new compare values in the comparator load registers are written to the compare registers eliminating the use of software to do this.

The compare load feature is intended to be used in variable frequency PWM mode. The COMP1 register determines the pulse width for the logic low part of OFLAG and COMP2 determines the pulse width for the logic high part of OFLAG. The period of the waveform is determined by the COMP1 and COMP2 values and the frequency of the primary clock source. See the following figure.

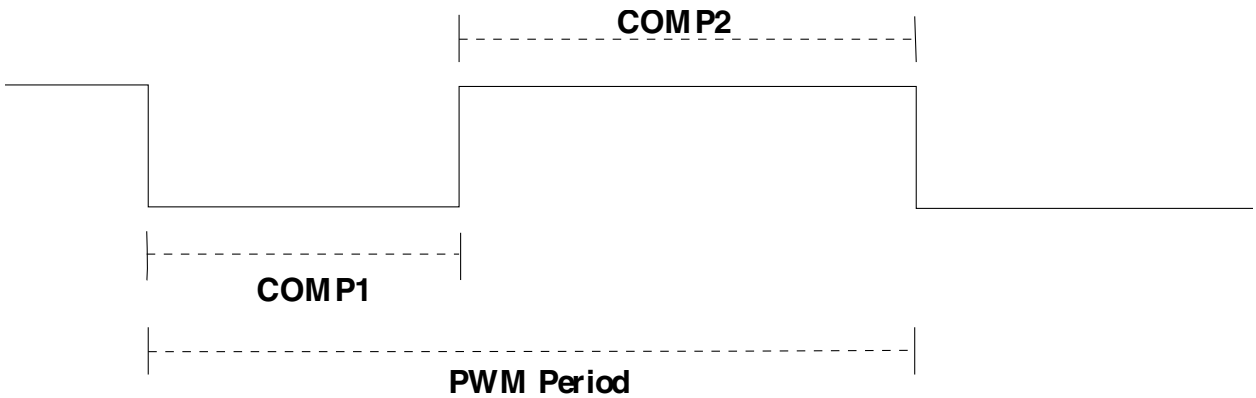


Figure 30-2. Variable PWM Waveform

Should we desire to update the duty cycle or period of the above waveform, we would need to update the COMP1 and COMP2 values using the compare load feature.

30.6.4 Usage of the Capture Register

The capture register stores a copy of the counter's value when an input edge (positive, negative, or both) is detected. After a capture event occurs, no further updating of the capture register will occur until the SCTRL[IEF] (input edge flag) is cleared by writing a zero to the SCTRL[IEF].

30.6.5 Functional Modes

The selected external count signals are sampled at the TMR's base clock rate and then run through a transition detector. The maximum count rate is one-half of the TMR's base clock rate. Internal clock sources can be used to clock the counters at the TMR's base clock rate.

If a counter is programmed to count to a specific value and then stop, the CTRL[CM] field is cleared when the count terminates.

30.6.5.1 Stop Mode

If CTRL[CM] is set to '000', the counter is inert. No counting will occur. Stop mode will also disable the interrupts caused by input transitions on a selected input pin.

30.6.5.2 Count Mode

If CTRL[CM] is set to '001', the counter will count the rising edges of the selected clock source. This mode is useful for generating periodic interrupts for timing purposes, or counting external events such as "widgets" on a conveyor belt passing a sensor. If the selected input is inverted by setting SCTRL[IPS] (input polarity select), then the negative edge of the selected external input signal is counted.

Example: 30.6.5.2.1 Count Pulses from External Source

```
//        (See Processor Expert PulseAccumulator bean.)
//    This example uses TMR1 to count pulse (actually counts rising edges of the pulse)
//      from an external source (QT3).
//
void Pulse_Init(void)
{
  /* TMR1_CTRL: CM=0,PCS=3,SCS=0,ONCE=0,LENGTH=0,DIR=0,Co_INIT=0,OM=0 */
  setReg(TMR1_CTRL,0x0600);           /* Set up mode */
  /* TMR1_SCTRL: TCF=0,TCFIE=0,TOF=0,TOFIE=0,IEF=0,IEFIE=0,IPS=0,INPUT=0,
                 Capture_Mode=0,MSTR=0,EEOF=0,VAL=0,FORCE=0,OPS=0,OEN=0 */
  setReg(TMR1_SCTRL,0x00);
  setReg(TMR1_CNTR,0x00);             /* Reset counter register */
  setReg(TMR1_LOAD,0x00);             /* Reset load register */
  setRegBitGroup(TMR1_CTRL,CM,0x01);  /* Run counter */
}
```

Example: 30.6.5.2.2 Generate Periodic Interrupt By Counting Internal Clocks

```
//         (See Processor Expert TimerInt bean.)
//   This example generates an interrupt every 100ms,
//     assuming the chip is operating at 60 MHz.
//
// It does this by using the IP_bus_clk divided by 128 as the counter clock source.
// The counter then counts to 46874 where it matches the COMP1 value.
// At that time an interrupt is generated, the counter is reloaded and
// the next COMP1 value is loaded from CMPLD1.
//
void TimerInt_Init(void)
{
  /* TMR0_CTRL: CM=0,PCS=0,SCS=0,ONCE=0,LENGTH=1,DIR=0,Co_INIT=0,OM=0 */
  setReg(TMR0_CTRL,0x20);             /* Stop all functions of the timer */
  /* TMR0_SCTRL: TCF=0,TCFIE=0,TOF=0,TOFIE=0,IEF=0,IEFIE=0,IPS=0,INPUT=0,
                 Capture_Mode=0,MSTR=0,EEOF=0,VAL=0,FORCE=0,OPS=0,OEN=0 */
  setReg(TMR0_SCTRL,0x00);
```

```
  setReg(TMR0_LOAD,0x00);             /* Reset load register */
  setReg(TMR0_COMP1,46874);           /* Set up compare 1 register */
  setReg(TMR0_CMPLD1,46874);          /* Also set the compare preload register */
  /* TMR0_CSCTRL: ??=0,??=0,??=0,??=0,??=0,??=0,??=0,??=0,TCF2EN=0,TCF1EN=1,
                  TCF2=0,TCF1=0,CL2=0,CL1=1 */
  setReg(TMR0_CSCTRL,0x41);           /* Enable compare 1 interrupt and */
                                       /*   compare 1 preload */
  setRegBitGroup(TMR0_CTRL,PCS,0xF); /* Primary Count Source to IP_bus_clk / 128 */
  setReg(TMR0_CNTR,0x00);             /* Reset counter register */
  setRegBitGroup(TMR0_CTRL,CM,0x01); /* Run counter */
}
```

30.6.5.3 Edge-Count Mode

If CTRL[CM] is set to '010', the counter will count both edges of the selected external clock source. This mode is useful for counting the changes in the external environment, such as a simple encoder wheel.

Example: 30.6.5.3.1 Count Both Edges of External Source Signal

```
//       (See Processor Expert PulseAccumulator bean.)
//  This example uses TMR1 to count pulse (actually counts rising edges of the pulse)
//    from an external source (QT3).
//
void Pulse_Init(void)
{
  /* TMR1_CTRL: CM=0,PCS=3,SCS=0,ONCE=0,LENGTH=0,DIR=0,Co_INIT=0,OM=0 */
  setReg(TMR1_CTRL,0x0600);           /* Set up mode */
  /* TMR1_SCTRL: TCF=0,TCFIE=0,TOF=0,TOFIE=0,IEF=0,IEFIE=0,IPS=0,INPUT=0,
                 Capture_Mode=0,MSTR=0,EEOF=0,VAL=0,FORCE=0,OPS=0,OEN=0 */
  setReg(TMR1_SCTRL,0x00);
  setReg(TMR1_CNTR,0x00);             /* Reset counter register */
  setReg(TMR1_LOAD,0x00);             /* Reset load register */
  setRegBitGroup(TMR1_CTRL,CM,0x02);  /* Run counter */
}
```

30.6.5.4 Gated-Count Mode

If CTRL[CM] is set to '011', the counter will count while the selected secondary input signal is high. This mode is used to time the duration of external events. If the selected input is inverted by setting SCTRL[IPS] (input polarity select), then the counter will count while the selected secondary input is low.

Example: 30.6.5.4.1 Capture Duration of External Pulse

```
//       (See Processor Expert PulseAccumulator bean.)
//   This example uses TMR1 to determine the duration of an external pulse.
//
//   The IP_bus clock is used as the primary counter. If the duration of the
//     external pulse is longer than 0.001 seconds one of the other IP_bus clock
//     dividers can be used. If the pulse duration is longer than 0.128 seconds
//     an external clock source will have to be used as the primary clock source.
//
void Pulse1_Init(void)
{
  /* TMR1_CTRL: CM=0,PCS=8,SCS=1,ONCE=0,LENGTH=0,DIR=0,Co_INIT=0,OM=0 */
  setReg(TMR1_CTRL,0x1080);             /* Set up mode */
  /* TMR1_SCTRL: TCF=0,TCFIE=0,TOF=0,TOFIE=0,IEF=0,IEFIE=0,IPS=0,INPUT=0,
                 Capture_Mode=0,MSTR=0,EEOF=0,VAL=0,FORCE=0,OPS=0,OEN=0 */
  setReg(TMR1_SCTRL,0x00);
  setReg(TMR1_CNTR,0x00);              /* Reset counter register */
  setReg(TMR1_LOAD,0x00);              /* Reset load register */
  setRegBitGroup(TMR1_CTRL,CM,0x03);  /* Run counter */
}
```

30.6.5.5 Quadrature-Count Mode

If CTRL[CM] is set to '100', the counter will decode the primary and secondary external inputs as quadrature encoded signals. Quadrature signals are usually generated by rotary or linear sensors used to monitor movement of motor shafts or mechanical equipment. The quadrature signals are square waves that are 90 degrees out of phase. The decoding of quadrature signal provides both count and direction information.

This figure shows a timing diagram illustrating the basic operation of a quadrature incremental position encoder.

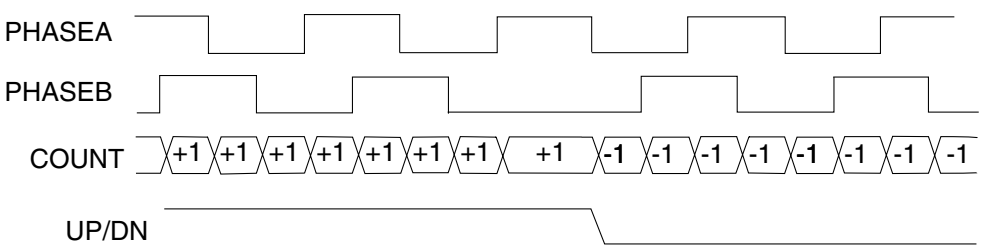


Figure 30-3. Quadrature Incremental Position Encoder

Example: 30.6.5.5.1 Quadrature Count Mode Example

```
//       (See Processor Expert PulseAccumulator bean.)
//  This example uses TMRA0 for counting states of a quadrature position encoder.
//
//  Timer input 0 is used as the primary count source (PHASEA).
//  Timer input 1 is used as the secondary count source (PHASEB).
//
void Pulse_Init(void)
{
```

```
  /* TMRA0_CTRL: CM=0,PCS=0,SCS=1,ONCE=0,LENGTH=0,DIR=0,Co_INIT=0,OM=0 */
  setReg(TMRC0_CTRL,0x80);              /* Set up mode */
  /* TMRA0_SCTRL: TCF=0,TCFIE=0,TOF=0,TOFIE=0,IEF=0,IEFIE=0,IPS=0,INPUT=0,
                Capture_Mode=0,MSTR=0,EEOF=0,VAL=0,FORCE=0,OPS=0,OEN=0 */
  setReg(TMRA0_SCTRL,0x00);
  setReg(TMRA0_CNTR,0x00);             /* Reset counter register */
  setReg(TMRA0_LOAD,0x00);             /* Reset load register */
  setReg(TMRA0_COMP1,0xFFFF);           /* Set up compare 1 register */
  setReg(TMRA0_COMP2,0x00);             /* Set up compare 2 register */
  /* TMRA0_CSCTRL: ??=0,??=0,??=0,??=0,??=0,??=0,??=0,??=0,
                  TCF2EN=0,TCF1EN=0,TCF2=0,TCF1=0,CL2=0,CL1=0 */
  setReg(TMRA0_CSCTRL,0x00);
  setRegBitGroup(TMRA0_CTRL,CM,0x04);  /* Run counter */
}
```

30.6.5.6 Quadrature-Count Mode with Index Input

As an extension to the quadrature count mode discussed in the previous paragraph, some rotary shafts have a HOME or INDEX indicator. This would be a third input to the timer that is used to reset the timer's counter.

In this example, channel 0 is used to decode the quadrature inputs, but it doesn't actually count. Because its upper and lower limits are both set to 0, its output is cascaded count up and count down signals each time the quadrature inputs indicate a change in count. Channel 1 works in cascaded count mode receiving its counting instructions from channel 0. When an input capture event occurs, channel 1 is programmed to reset its counter value. The channel 1 counter contains the position value for the shaft.

Example: 30.6.5.6.1 Quadrature Count Mode with Index Input Example

```
//       (See Processor Expert PulseAccumulator bean.)
//  This example uses TMR0and TMR1 for counting states of a quadrature position encoder.
//
// Timer input 0 is used as the primary count source (PHASEA).
// Timer input 1 is used as the secondary count source (PHASEB).
// Timer input 2 is used as the index input source (INDEX).
//
void Pulse_Init(void)
{
  /* TMR0_CTRL: CM=0,PCS=0,SCS=1,ONCE=0,LENGTH=1,DIR=0,Co_INIT=0,OM=0 */
  setReg(TMR0_CTRL,0xA0);              /* Set up mode */
  /* TMR0_SCTRL: TCF=0,TCFIE=0,TOF=0,TOFIE=0,IEF=0,IEFIE=0,IPS=0,INPUT=0,
                Capture_Mode=0,MSTR=0,EEOF=0,VAL=0,FORCE=0,OPS=0,OEN=0 */
  setReg(TMR0_SCTRL,0x00);
  setReg(TMR0_CNTR,0x00);             /* Reset counter register */
  setReg(TMR0_LOAD,0x00);             /* Reset load register */
  setReg(TMR0_COMP1,0x00);           /* Set up compare 1 register */
  setReg(TMR0_COMP2,0x00);             /* Set up compare 2 register */
  /* TMR0_CSCTRL: ??=0,??=0,??=0,??=0,??=0,??=0,??=0,??=0,
                  TCF2EN=0,TCF1EN=0,TCF2=0,TCF1=0,CL2=0,CL1=0 */
  setReg(TMR0_CSCTRL,0x00);
/* TMR1_CTRL: CM=7,PCS=100,SCS=2,ONCE=0,LENGTH=0,DIR=0,Co_INIT=0,OM=0 */
  setReg(TMR1_CTRL,0xEA00);            /* Set up capture edge */
/* TMR1_SCTRL: Capture_Mode=10 */
```

```
  setReg(TMR1_SCTRL,0x0080);
/* TMR1_CSTRL: ROC=1 */
  setReg(TMR1_CTRL,0x0800);           /* Set up reload on capture */
setRegBitGroup(TMR0_CTRL,CM,0x04);  /* Run counter */
}
```

30.6.5.7 Signed-Count Mode

If CTRL[CM] is set to '101', the counter counts the primary clock source while the selected secondary source provides the selected count direction (up/down).

Example: 30.6.5.7.1 Signed Count Mode Example

```
//       (See Processor Expert PulseAccumulator bean.)
//  This example uses TMR0 for signed mode counting.
//
//  Timer input 2 is used as the primary count source.
//  Timer input 1 is used to determine the count direction.
//
void Pulse_Init(void)
{
  /* TMR0_CTRL: CM=0,PCS=2,SCS=1,ONCE=0,LENGTH=0,DIR=0,Co_INIT=0,OM=0 */
  setReg(TMR0_CTRL,0x0480);             /* Set up mode */
  /* TMR0_SCTRL: TCF=0,TCFIE=0,TOF=0,TOFIE=1,IEF=0,IEFIE=0,IPS=0,INPUT=0,
                 Capture_Mode=0,MSTR=0,EEOF=0,VAL=0,FORCE=0,OPS=0,OEN=0 */
  setReg(TMR0_SCTRL,0x1000);
  setReg(TMR0_CNTR,0x00);             /* Reset counter register */
  setReg(TMR0_LOAD,0x00);             /* Reset load register */
  setRegBitGroup(TMR0_CTRL,CM,0x05);  /* Run counter */
}
```

30.6.5.8 Triggered-Count Mode 1

If CSCTRL[TCI] is clear and CTRL[CM] is set to '110', the counter will begin counting the primary clock source after a positive transition (negative edge if SCTRL[IPS]=1) of the secondary input occurs. The counting will continue until a compare event occurs or another positive input transition is detected. If a second input transition occurs before a terminal count is reached, counting will stop and SCTRL[TCF] (timer compare flag) will be set. Subsequent secondary input transitions will continue to restart and stop the counting until a compare event occurs.

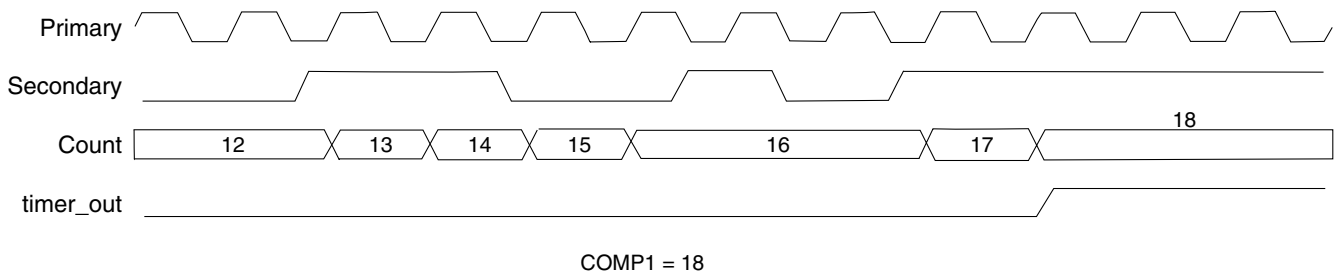


Figure 30-4. Triggered Count Mode 1 (CTRL[LENGTH]=0)

Example: 30.6.5.8.1 Triggered Count Mode 1 Example

```
//        (See Processor Expert PulseAccumulator bean.)
//   This example uses TMR1 for triggered mode counting.
//
//   Timer input 3 is used as the primary count source.
//   Timer input 2 is used for the trigger input.
//
void Pulse_Init(void)
{
  /* TMR1_CTRL: CM=0,PCS=3,SCS=2,ONCE=0,LENGTH=0,DIR=0,Co_INIT=0,OM=0 */
  setReg(TMR1_CTRL,0x0700);            /* Set up mode */
  /* TMR1_SCTRL: TCF=0,TCFIE=0,TOF=0,TOFIE=1,IEF=0,IEFIE=0,IPS=0,INPUT=0,
                 Capture_Mode=0,MSTR=0,EEOF=0,VAL=0,FORCE=0,OPS=0,OEN=0 */
  setReg(TMR1_SCTRL,0x1000);
  setReg(TMR1_CNTR,0x00);              /* Reset counter register */
  setReg(TMR1_LOAD,0x00);              /* Reset load register */
  setReg(TMR1_COMP1,0x0012);            /* Set up compare 1 register */
  /* TMR1_CSCTRL: ??=0,??=0,??=0,??=0,??=0,??=0,??=0,??=0,
                  TCF2EN=0,TCF1EN=0,TCF2=0,TCF1=0,CL2=0,CL1=0 */
  setReg(TMR1_CSCTRL,0x00);
  setRegBitGroup(TMR1_CTRL,CM,0x06);    /* Run counter */
}
```

30.6.5.9 Triggered-Count Mode 2

If CSCTRL[TCI] is set and CTRL[CM] is set to '110', the counter will begin counting the primary clock source after a positive transition (negative edge if SCTRL[IPS]=1) of the secondary input occurs. The counting will continue until a compare event occurs or another positive input transition is detected. If a second input transition occurs before a terminal count was reached, the counter will reload and continue counting. When CSCTRL[TCI] is set, the OFLAG output mode, CTRL[OUTMODE], should probably be set to '101' (cleared on init, set on compare) to ensure the output will be in a known state after the second input transition and subsequent reload takes place.

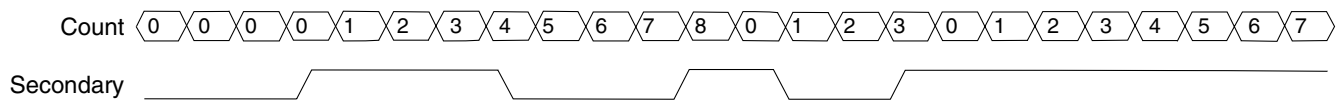


Figure 30-5. Triggered Count Mode 2 (CTRL[LENGTH]=0)

Example: 30.6.5.9.1 Triggered Count Mode 2Example

```
//        (See Processor Expert PulseAccumulator bean.)
//   This example uses TMR1 for triggered mode counting.
//
//   Timer input 3 is used as the primary count source.
//   Timer input 2 is used for the trigger input.
//
void Pulse_Init(void)
{
  /* TMR1_CTRL: CM=0,PCS=3,SCS=2,ONCE=0,LENGTH=0,DIR=0,Co_INIT=0,OM=0 */
  setReg(TMR1_CTRL,0x0700);           /* Set up mode */
  /* TMR1_SCTRL: TCF=0,TCFIE=0,TOF=0,TOFIE=1,IEF=0,IEFIE=0,IPS=0,INPUT=0,
                 Capture_Mode=0,MSTR=0,EEOF=0,VAL=0,FORCE=0,OPS=0,OEN=0 */
  setReg(TMR1_SCTRL,0x1000);

  setReg(TMR1_CNTR,0x00);             /* Reset counter register */
  setReg(TMR1_LOAD,0x00);             /* Reset load register */
  setReg(TMR1_COMP1,0x0012);           /* Set up compare 1 register */
  /* TMR1_CSCTRL: ??=0,??=0,??=0,??=0,??=0,??=0,??=0,??=0,
                  TCF2EN=0,TCF1EN=0,TCF2=0,TCF1=0,CL2=0,CL1=0 */
  setReg(TMR1_CSCTRL,0x00);
  setRegBitGroup(TMR1_CTRL,CM,0x06); /* Run counter */
}
```

30.6.5.10 One-Shot Mode

If CTRL[CM] is set to '110', and the counter is set to reinitialize at a compare event (CTRL[LENGTH]=1),and CTRL[OUTMODE] is set to '101' (cleared on init, set on compare), the counter works in a one-shot mode. An external event causes the counter to count, and when the terminal count is reached, the output is asserted. This delayed output can be used to provide timing delays.

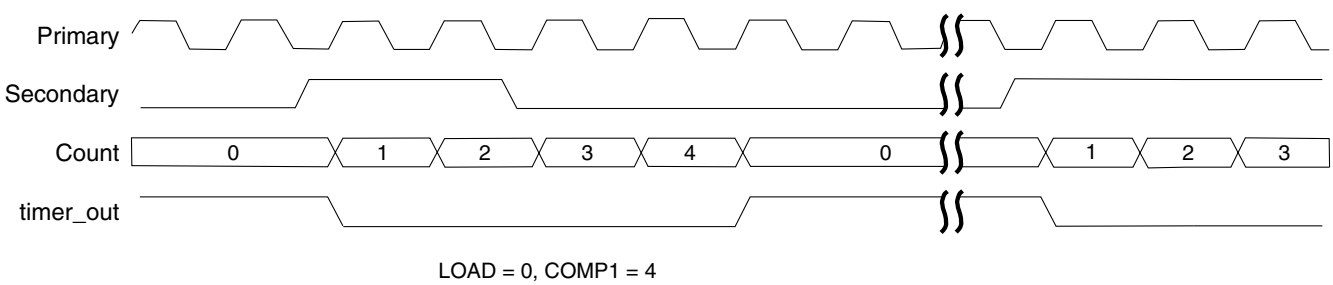


Figure 30-6. One-Shot Mode (CTRL[LENGTH]=1)

Example: 30.6.5.10.1 One-Shot Mode Example

```
//   (See Processor Expert PulseAccumulator bean.)
//  This example uses TMR1 for one-shot mode counting.
//
//  Timer input 3 is used as the primary count source.
//  Timer input 2 is used for the trigger input.
//
void Pulse_Init(void)
{
  /* TMR1_CTRL: CM=0,PCS=3,SCS=2,ONCE=0,LENGTH=0,DIR=0,Co_INIT=0,OM=5 */
  setReg(TMR1_CTRL,0x0725);           /* Set up mode */
  /* TMR1_SCTRL: TCF=0,TCFIE=0,TOF=0,TOFIE=1,IEF=0,IEFIE=0,IPS=0,INPUT=0,
              Capture_Mode=0,MSTR=0,EEOF=0,VAL=0,FORCE=0,OPS=0,OEN=0 */
  setReg(TMR1_SCTRL,0x1000);
  setReg(TMR1_CNTR,0x00);             /* Reset counter register */
  setReg(TMR1_LOAD,0x00);             /* Reset load register */
  setReg(TMR1_COMP1,0x0004);           /* Set up compare 1 register */
  /* TMR1_CSCTRL: ??=0,??=0,??=0,??=0,??=0,??=0,??=0,??=0,
                  TCF2EN=0,TCF1EN=0,TCF2=0,TCF1=0,CL2=0,CL1=0 */
  setReg(TMR1_CSCTRL,0x00);
  setRegBitGroup(TMR1_CTRL,CM,0x06);  /* Run counter */
}
```

30.6.5.11 Cascade-Count Mode

If CTRL[CM] is set to '111', the counter's input is connected to the output of another selected counter. The counter will count up and down as compare events occur in the selected source counter. This cascade or daisy-chained mode enables multiple counters to be cascaded to yield longer counter lengths. When operating in cascade mode, a special high-speed signal path is used between modules rather than the OFLAG output signal. If the selected source counter is counting up and it experiences a compare event, the counter will be incremented. If the selected source counter is counting down and it experiences a compare event, the counter will be decremented.

Up to four counters may be cascaded to create a 64-bit wide synchronous counter. Check the data sheet to see if there are any frequency limits for cascaded counting mode.

Whenever any counter is read within a counter module, all of the counters' values within the module are captured in their respective hold registers. This action supports the reading of a cascaded counter chain. First read any counter of a cascaded counter chain, then read the hold registers of the other counters in the chain. The cascaded counter mode is synchronous.

Note

It is possible to connect counters together by using the other (non-cascade) counter modes and selecting the outputs of other counters as a clock source. In this case, the counters are operating in a ripple mode, where higher order counters will transition a clock later than a purely synchronous design.

Example: 30.6.5.11.1 Generate Periodic Interrupt Cascading Two Counters

```
//    (See Processor Expert TimerInt bean.)
// This example generates an interrupt every 30 seconds,
//    assuming the chip is operating at 60 MHz.
//
// To do this, counter 2 is used to count 60,000 IP_bus clocks, which means it
//    will compare and reload every 0.001 seconds.
// Counter 3 is cascaded and used to count the 0.001 second ticks and
//    generate the desired interrupt interval.
//
void TimerInt_Init(void)
{
// Set counter 2 to count IP_bus clocks
   /* TMR2_CTRL: CM=0,PCS=8,SCS=0,ONCE=0,LENGTH=1,DIR=0,Co_INIT=0,OM=0 */
   setReg(TMR2_CTRL,0x1020);           /* Stop all functions of the timer */
// Set counter 3 as cascaded and to count counter 2 outputs
  /* TMR3_CTRL: CM=7,PCS=6,SCS=0,ONCE=0,LENGTH=1,DIR=0,Co_INIT=0,OM=0 */
  setReg(TMR3_CTRL,0xEC20);            /* Set up cascade counter mode */
  /* TMR3_SCTRL: TCF=0,TCFIE=0,TOF=0,TOFIE=0,IEF=0,IEFIE=0,IPS=0,INPUT=0,
                 Capture_Mode=0,MSTR=0,EEOF=0,VAL=0,FORCE=0,OPS=0,OEN=0 */
  setReg(TMR3_SCTRL,0x00);
  /* TMR2_SCTRL: TCF=0,TCFIE=0,TOF=0,TOFIE=0,IEF=0,IEFIE=0,IPS=0,INPUT=0,
                 Capture_Mode=0,MSTR=0,EEOF=0,VAL=0,FORCE=0,OPS=0,OEN=0 */
  setReg(TMR2_SCTRL,0x00);
  setReg(TMR3_CNTR,0x00);             /* Reset counter register */
  setReg(TMR2_CNTR,0x00);
  setReg(TMR3_LOAD,0x00);             /* Reset load register */
  setReg(TMR2_LOAD,0x00);
  setReg(TMR3_COMP1, 30000);           /* milliseconds in 30 seconds */
  setReg(TMR3_CMPLD1,30000);
  setReg(TMR2_COMP1, 60000);          // Set to cycle every milisecond
  setReg(TMR2_CMPLD1,60000);
  /* TMR3_CSCTRL: ??=0,??=0,??=0,??=0,??=0,??=0,??=0,??=0,
                  TCF2EN=0,TCF1EN=1,TCF2=0,TCF1=0,CL2=0,CL1=1 */
  setReg(TMR3_CSCTRL,0x41);           /* Enable compare 1 interrupt and */
                                       /* compare 1 preload              */
  /* TMR2_CSCTRL: ??=0,??=0,??=0,??=0,??=0,??=0,??=0,??=0,
                  TCF2EN=0,TCF1EN=0,TCF2=0,TCF1=0,CL2=0,CL1=1 */
  setReg(TMR2_CSCTRL,0x01);           /* Enable Compare 1 preload */
  setRegBitGroup(TMR2_CTRL,CM,0x01);  /* Run counter */
}
```

30.6.5.12 Pulse-Output Mode

If CTRL[CM]=001, and CTRL[OUTMODE] is set to 111' (gated clock output), and CTRL[ONCE] is set, then the counter will output a pulse stream of pulses that has the same frequency of the selected clock source, and the number of output pulses is equal to the compare value minus the init value.

Note

This does not work if CTRL[PCS] is set to 1000 (IP_bus/1).

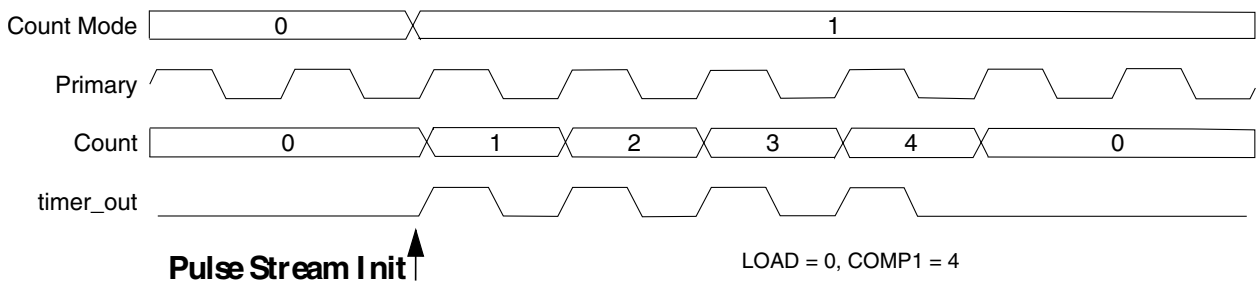


Figure 30-7. Pulse Output Mode

Example: 30.6.5.12.1 Pulse Outputs Using Two Counters

```
//     (See Processor Expert PulseStream bean.)
// This example generates six 10ms pulses, from QT1 output.
// Assuming the chip is operating at 60 MHz.
//
// To do this, timer 3 is used to generate a clock with a period of 10ms.
//
// Timer 1 is used to gate these clocks and count the number of pulses that have
//  been generated.
//
void PulseStream_Init(void)
{
// Select IP_bus_clk/16 as the clock source for Timer 3
  /* TMR3_CTRL: CM=0,PCS=0x0C,SCS=0,ONCE=0,LENGTH=1,DIR=0,Co_INIT=0,OM=3 */
   setReg(TMR3_CTRL,0x1823);             /* Set up mode */
  /* TMR3_SCTRL: TCF=0,TCFIE=0,TOF=0,TOFIE=0,IEF=0,IEFIE=0,IPS=0,INPUT=0,
                 Capture_Mode=0,MSTR=0,EEOF=0,VAL=0,FORCE=0,OPS=0,OEN=0 */
   setReg(TMR3_SCTRL,0x00);
   setReg(TMR3_LOAD,0x00);              /* Reset load register */
   setReg(TMR3_COMP1,37500);            /* (16 * 37500 ) / 60e6 = 0.01 sec */
 /* TMR3_CSCTRL: ??=0,??=0,??=0,??=0,??=0,??=0,??=0,??=0,
                   TCF2EN=0,TCF1EN=0,TCF2=0,TCF1=0,CL2=0,CL1=0 */
  setReg(TMR3_CSCTRL,0x00);             /* Set up comparator control register */



// Timer 3 output is the clock source for this timer.
  /* TMR1_CTRL: CM=0,PCS=7,SCS=0,ONCE=1,LENGTH=1,DIR=0,Co_INIT=0,OM=7 */
  setReg(TMR1_CTRL,0x0E67);            /* Set up mode */
  /* TMR1_SCTRL: TCF=0,TCFIE=0,TOF=0,TOFIE=0,IEF=0,IEFIE=0,IPS=0,INPUT=0,
                 Capture_Mode=0,MSTR=0,EEOF=0,VAL=0,FORCE=0,OPS=0,OEN=1 */
   setReg(TMR1_SCTRL,0x01);
```

```
   setReg(TMR1_CNTR,0x00);             /* Reset counter register */
   setReg(TMR1_LOAD,0x00);             /* Reset load register */
   setReg(TMR1_COMP1,0x04);             /* Set up compare 1 register */

// set to interrupt after the last pulse
  /* TMR1_CSCTRL: ??=0,??=0,??=0,??=0,??=0,??=0,??=0,??=0,
                  TCF2EN=0,TCF1EN=1,TCF2=0,TCF1=0,CL2=0,CL1=0 */
  setReg(TMR1_CSCTRL,0x40);             /* Set up comparator control register */
// Finally, start the counters running
  setReg(TMR3_CNTR,0);                  /* Reset counter */
  setRegBitGroup(TMR3_CTRL,CM,0x01);    /* Run source clock counter */
  setRegBitGroup(TMR1_CTRL,CM,0x01);    /* Run counter */
}
```

30.6.5.13 Fixed-Frequency PWM Mode

If CTRL[CM]=001, count through roll-over (CTRL[LENGTH]=0), continuous count (CTRL[ONCE]=0) and CTRL[OUTMODE] is '110' (set on compare, cleared on counter roll-over), then the counter output yields a pulse-width modulated (PWM) signal with a frequency equal to the count clock frequency divided by 65,536 and a pulse-width duty cycle equal to the compare value divided by 65,536. This mode of operation is often used to drive PWM amplifiers used to power inverters.

Example: 30.6.5.13.1 Fixed-Frequency PWM Mode Example

```
//    (See Processor Expert PWM bean.)
// This example uses TMR0 for Fixed-Frequency PWM mode timing.
//
// The timer will count IP_bus clocks continuously until it rolls over.
// This results in a PWM period of 65536 / 60e6 = 1092.267 usec
//
// Initially, an output pulse width of 25 usec is generated ( 1500 / 60e6 )
//   giving a PWM ratio of 1500 / 65536 = 2.289%
// This pulse width can be changed by changing the COMP1 register value (using CMPLD1).
//
void PWM1_Init(void)
{
  setReg(TMR0_CNTR,0);                 /* Reset counter */
  /* TMR0_SCTRL: TCF=0,TCFIE=0,TOF=0,TOFIE=0,IEF=0,IEFIE=0,IPS=0,INPUT=0,
                 Capture_Mode=0,MSTR=0,EEOF=0,VAL=0,FORCE=1,OPS=0,OEN=1 */
  setReg(TMR0_SCTRL,0x05);              /* Enable output */
  setReg(TMR0_COMP1,1500);          /* Store initial value to the duty-compare register */
  /* TMR0_CTRL: CM=1,PCS=8,SCS=0,ONCE=0,LENGTH=0,DIR=0,Co_INIT=0,OM=6 */
  setReg(TMR0_CTRL,0x3006);       /* Run counter */
}
```

30.6.5.14 Variable-Frequency PWM Mode

If CTRL[CM]=001, count until compare (CTRL[LENGTH]=1), continuous count (CTRL[ONCE] = 0) and CTRL[OUTMODE] is '100' (toggle OFLAG and alternate compare registers), then the counter output yields a pulse-width modulated (PWM) signal whose frequency and pulse width is determined by the values programmed into the COMP1 and COMP2 registers, and the input clock frequency. This method of PWM generation has the advantage of allowing almost any desired PWM frequency and/or constant on or off periods. This mode of operation is often used to drive PWM amplifiers used to power inverters. The CMPLD1 and CMPLD2 registers are especially useful for this mode, as they allow the programmer time to calculate values for the next PWM cycle while the PWM current cycle is underway.

To set up the TMR to run in variable frequency PWM mode with compare preload please use the following setup for the specific counter you would like to use. When performing the setup, update the TMR_CTRL register last because the counter will start counting if the count mode is changed to any value other than 000 (assuming the primary count source is already active).

Timer Control Register (CTRL)

- CM=001 (count rising edges of primary source)
- PCS=1000 (IP bus clock for best granularity for waveform timing)
- SCS=Any (ignored in this mode)
- ONCE=0 (want to count repeatedly)
- LENGTH=1 (want to count until compare value is reached and re-initialize counter register)
- DIR=Any (user's choice. The compare register values must be chosen carefully to account for things like roll-under, etc.)
- COINIT=0 (user can set this if they need this function)
- OUTMODE=100 (toggle OFLAG output using alternating compare registers)

Timer Status and Control Register (SCTRL)

- OEN = 1 (output enable to allow OFLAG output to be put on an external pin. Set this bit as needed.)

- OPS = Any (user's choice)
- Make sure the rest of the bits are cleared for this register. We will enable interrupts in the comparator status and control register instead of in this register.

Comparator Status and Control Register (CSCTRL)

- TCF2EN=1 (allow interrupt to be issued when CSCTRL[TCF2] is set)
- TCF1EN=0 (do not allow interrupt to be issued when CSCTRL[TCF1] is set)
- TCF1=0 (clear timer compare 1 interrupt source flag. This is set when counter register equals compare register 1 value and OFLAG is low)
- TCF2=0 (clear timer compare 2 interrupt source flag. This is set when counter register equals compare register 2 value and OFLAG is high)
- CL1=10 (load compare register when CSCTRL[TCF2] is asserted)
- CL2=01 (load compare register when CSCTRL[TCF1] is asserted)

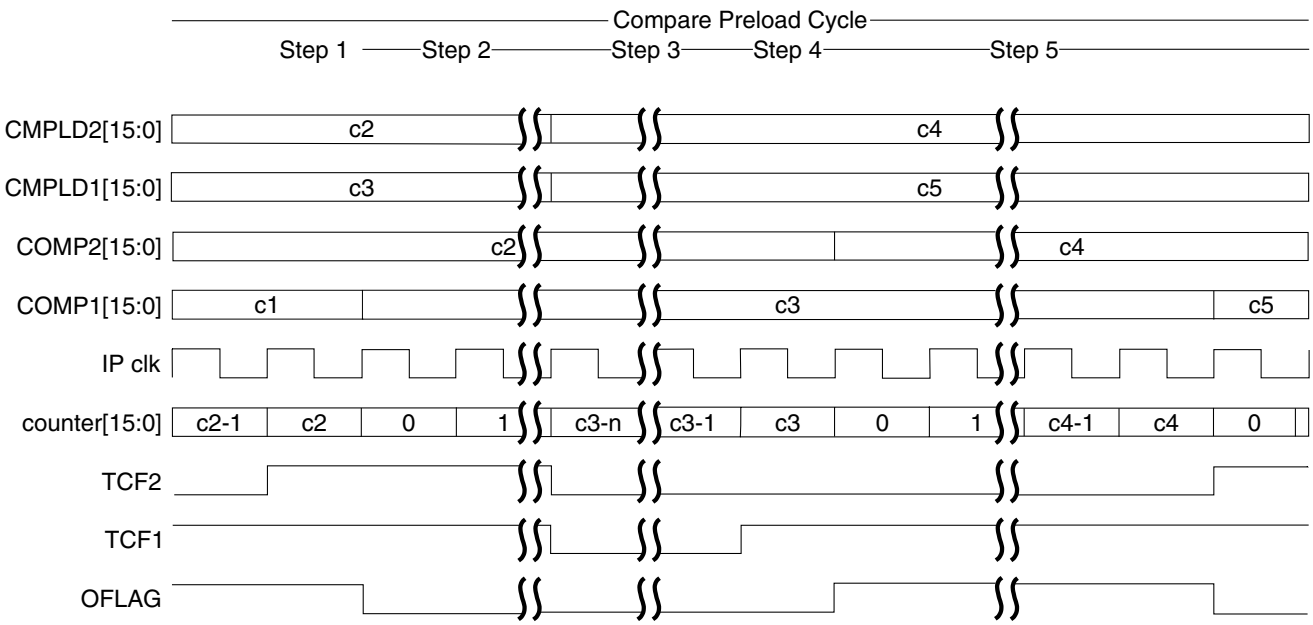
Interrupt Service Routines

To service the CSCTRL[TCF2] interrupts generated by the timer, the interrupt controller must be configured to enable the interrupts for the particular timer being used. Additionally the user will need to write an interrupt service routine to do at a minimum the following:

- Clear CSCTRL[TCF2] and CSCTRL[TCF1] flags.
- Calculate and write new values for both CMPLD1 and CMPLD2.

Timing

This figure contains the timing for using the compare preload feature. The compare preload cycle begins with a compare event on COMP2 causing CSCTRL[TCF2] to be asserted. COMP1 is loaded with the value in the CMPLD1 (c3) one IP bus clock later. In addition an interrupt is asserted by the timer and the interrupt service routine is executed during which both comparator load registers are updated with new values (c4 and c5). When CSCTRL[TCF1] is asserted, COMP2 is loaded with the value in CMPLD2 (c4). And on the subsequent CSCTRL[TCF2] event, COMP1 is loaded with the value in CMPLD1 (c5). The cycle starts over again as an interrupt is asserted and the interrupt service routine clears CSCTRL[TCF1] and CSCTRL[TCF2] and calculates new values for CMPLD1 and CMPLD2.



Step 1-- CNTR matches COMP2 value. CSCTRL[TCF2] is asserted and an interrupt request is generated.

Step 2-- One clock later, OFLAG toggles, CMPLD1 is copied to COMP1, LOAD is copied to CNTR, the counter starts counting.

Step 3-- The interrupt service routine clears CSCTRL[TCF1] and CSCTRL[TCF2] and the ISR loads CMPLD1 and CMPLD2 with the values for the next cycle. The counter continues counting until CNTR matches COMP1.

Step 4-- CSCTRL[TCF1] is asserted. One clock later, OFLAG toggles, CMPLD2 is copied to COMP2, LOAD is copied to CNTR and the counter starts counting.

Step 5--The counter continues counting until CNTR matches COMP2.

Figure 30-8. Compare Load Timing

Example: 30.6.5.14.1 Variable Frequency PWM Mode

```
//    (See Processor Expert PPG [Programmable Pulse Generator] bean.)
// This example starts with an 11 msec with a 31 msec cycle.
// Assuming the chip is operating at 60 MHz, the timer use IP_bus_clk/32 as its
//   clock source.
//
// Initial pulse period: 60e6/32 clocks/sec * 31 ms = 58125 total clocks in period
// Initial pulse width: 60e6/32 clocks/sec * 11 ms = 20625 clocks in pulse
//
//
// Once the initial values of COMP1/CMPLD1 and COMP2/CMPLD2 are set the pulse width
//   can be varied by load new values of CMPLD1 and CMPLD2 on each compare interrupt.
//    (See Usage of Compare Load Registers.)
//
void PPG1_Init(void)
{
  setReg(TMR0_LOAD,0);                /* Clear load register */
```

```
  setReg(TMR0_CNTR,0);                /* Clear counter */
  /* TMR0_SCTRL: TCF=0,TCFIE=0,TOF=0,TOFIE=0,IEF=0,IEFIE=0,IPS=0,INPUT=0,
                 Capture_Mode=0,MSTR=0,EEOF=0,VAL=0,FORCE=1,OPS=0,OEN=1 */
  setReg(TMR0_SCTRL,5);                /* Set Status and Control Register */
// Set compare preload operation and enable an interrupt on compare2 events.
  /* TMR0_CSCTRL: TCF2EN=1,TCF1EN=0,TCF2=0,TCF1=0,CL21=0,CL20=1,CL11=1,CL10=0 */
  setReg(TMR0_CSCTRL,0x86);          /* Set Comparator Status and Control Register */

  setReg(TMR0_COMP1,20625);            /* Set the pulse width of the off time */
  setReg(TMR0_CMPLD1,20625);          /* Set the pulse width of the off time */
  setReg(TMR0_COMP2,58125-20625);      /* Set the pulse width of the on time */
  setReg(TMR0_CMPLD2,58125-20625);    /* Set the pulse width of the on time */
  /* TMR0_CTRL: CM=1,PCS=0xD,SCS=0,ONCE=0,LENGTH=1,DIR=0,Co_INIT=0,OM=4 */
  setRegBits(TMR0_CTRL,0x3A24);        /* Set variable PWM mode and run counter */
}
```

30.7 Resets

30.7.1 General

The TMR module can be reset only by the RST_B signal. This forces all registers to their reset state and clears the OFLAG signal if it is asserted. The counter will be turned off until the settings in the control register are changed.

Table 30-1. Reset Summary

Reset	Priority	Source	Characteristics
RST_B	n/a	Hardware Reset	Full System Reset

30.8 Clocks

30.8.1 General

The timer only receives the IP bus clock (system clock).

30.9 Interrupts

30.9.1 General

The TMR module can generate 20 interrupts, Five for each of the four counters/channels.

Table 30-2. Interrupt Summary

Core Interrupt	Interrupt	Description
TMR Channel 0	TMR0_COMP_IRQ_B	Compare Interrupt Request for Timer Channel 0
	TMR0_COMP1_IRQ_B	Compare 1 Interrupt Request for Timer Channel 0
	TMR0_COMP2_IRQ_B	Compare 2 Interrupt Request for Timer Channel 0
	TMR0_OVF_IRQ_B	Overflow Interrupt Request for Timer Channel 0
	TMR0_EDGE_IRQ_B	Input Edge Interrupt Request for Timer Channel 0
TMR Channel 1	TMR1_COMP_IRQ_B	Compare Interrupt Request for Timer Channel 1
	TMR1_COMP1_IRQ_B	Compare 1 Interrupt Request for Timer Channel 1
	TMR1_COMP2_IRQ_B	Compare 2 Interrupt Request for Timer Channel 1
	TMR1_OVF_IRQ_B	Overflow Interrupt Request for Timer Channel 1
	TMR1_EDGE_IRQ_B	Input Edge Interrupt Request for Timer Channel 1
TMR Channel 2	TMR2_COMP_IRQ_B	Compare Interrupt Request for Timer Channel 2
	TMR2_COMP1_IRQ_B	Compare 1 Interrupt Request for Timer Channel 2
	TMR2_COMP2_IRQ_B	Compare 2 Interrupt Request for Timer Channel 2
	TMR2_OVF_IRQ_B	Overflow Interrupt Request for Timer Channel 2
	TMR2_EDGE_IRQ_B	Input Edge Interrupt Request for Timer Channel 2
TMR Channel 3	TMR3_COMP_IRQ_B	Compare Interrupt Request for Timer Channel 3
	TMR3_COMP1_IRQ_B	Compare 1 Interrupt Request for Timer Channel 3
	TMR3_COMP2_IRQ_B	Compare 2 Interrupt Request for Timer Channel 3
	TMR3_OVF_IRQ_B	Overflow Interrupt Request for Timer Channel 3
	TMR3_EDGE_IRQ_B	Input Edge Interrupt Request for Timer Channel 3

30.9.2 Description of Interrupt Operation

30.9.2.1 Timer Compare Interrupts

These interrupts are generated when a successful compare occurs between a counter and it's compare registers while SCTRL[TCFIE] is set. These interrupts are cleared by writing a zero to the appropriate SCTRL[TCF].

When a timer compare interrupt is set in TMR_SCTRL and the Compare Load registers are available, one of the following two interrupts will also be asserted.

30.9.2.1.1 Timer Compare 1 Interrupts (Available with Compare Load Feature)

These interrupts are generated when a successful compare occurs between a counter and its COMP1 register while CSCTRL[TCF1EN] is set. These interrupts are cleared by writing a zero to the appropriate CSCTRL[TCF1].

30.9.2.1.2 Timer Compare 2 Interrupts (Available with Compare Load Feature)

These interrupts are generated when a successful compare occurs between a counter and its COMP2 register while CSCTRL[TCF2EN] is set. These interrupts are cleared by writing a zero to the appropriate CSCTRL[TCF2].

30.9.2.2 Timer Overflow Interrupts

These interrupts are generated when a counter rolls over its maximum value while SCTRL[TOFIE] is set. These interrupts are cleared by writing zero to the appropriate SCTRL[TOF].

30.9.2.3 Timer Input Edge Interrupts

These interrupts are generated by a transition of the input signal (either positive or negative depending on IPS setting) while SCTRL[IEFIE] is set. These interrupts are cleared by writing a zero to the appropriate SCTRL[IEF].

30.10 DMA

The TMR module can generate twelve DMA requests: three for each of the four counters/channels. Refer to the following table.

Table 30-3. DMA Summary

DMA Request	DMA Enable	Description
Channels 0-3	DMA[IEFDE]	CAPT contains a value
	DMA[CMPLD1DE]	CMPLD1 needs an update
	DMA[CMPLD2DE]	CMPLD2 needs an update

Chapter 31
Periodic Interrupt Timer (PIT)

31.1 Introduction

The programmable interval timer module (PIT) contains clock select logic, a 16-bit up counter, a modulo register, and a control register. The modulo and control registers are read/writable. The counter is read only.

The modulo register is loaded with a value to count to and the prescaler is set to determine the counting rate. When enabled, the counter counts up to the modulo value and set a flag (and an interrupt request if enabled), reset to 0x0000, and resume counting.

31.1.1 Features

The PIT module design includes these distinctive features:

- 16-bit counter/timer.
- Programmable count modulo.
- Up to 4 selectable clock sources.
- Maximum count rate equal to clocking rate.
- Slave mode allows synchronization of multiple PIT count enables.

31.1.2 Modes of Operation

The PIT module design operates in only a single mode of operation: functional mode.

31.1.3 Block Diagram

The following figure shows the PIT block diagram.

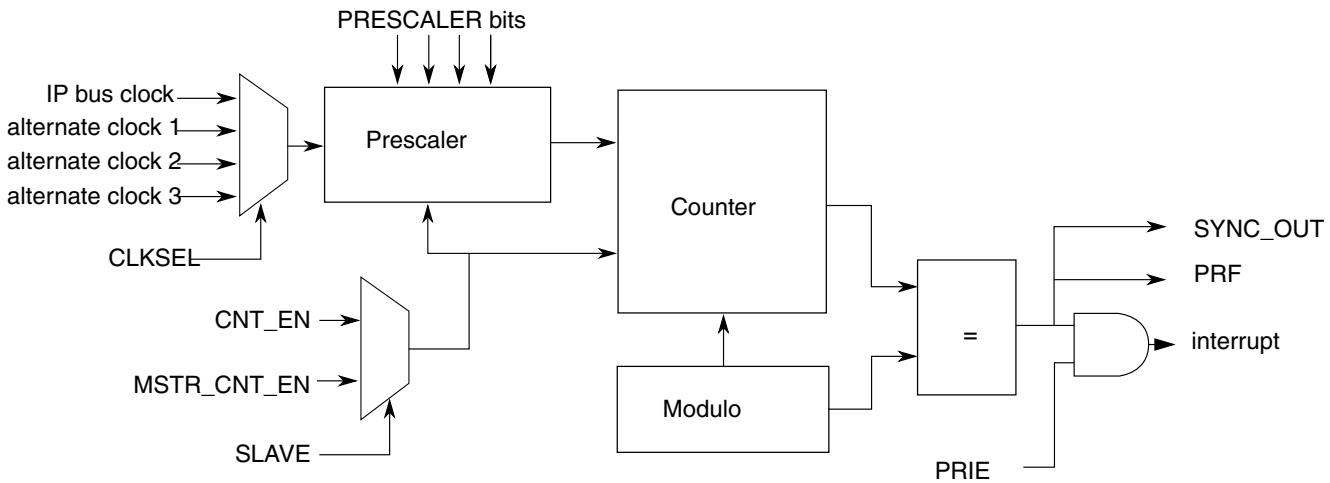


Figure 31-1. Programmable Interval Timer Block Diagram

31.2 Memory Map and Registers

The base address of the PIT module differs from chip to chip. The following descriptions identify the locations of memory mapped registers in relation to the base address.

The address of a register is the sum of a base address and an address offset. The base address is defined at the core level and the address offset is defined at the module level.

PIT memory map

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
E100	PIT Control Register (PIT0_CTRL)	16	R/W	0000h	31.2.1/777
E101	PIT Modulo Register (PIT0_MOD)	16	R/W	0000h	31.2.2/778
E102	PIT Counter Register (PIT0_CNTR)	16	R	0000h	31.2.3/779
E110	PIT Control Register (PIT1_CTRL)	16	R/W	0000h	31.2.1/777
E111	PIT Modulo Register (PIT1_MOD)	16	R/W	0000h	31.2.2/778
E112	PIT Counter Register (PIT1_CNTR)	16	R	0000h	31.2.3/779

31.2.1 PIT Control Register (PITx_CTRL)

Address: Base address + 0h offset

Bit	15	14	13	12	11	10	9	8
Read	SLAVE	0					CLKSEL	
Write								
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
Read	0	PRESCALER				PRF	PRIE	CNT_EN
Write								
Reset	0	0	0	0	0	0	0	0

PITx_CTRL field descriptions

Field	Description
15 SLAVE	This field is used to place this PIT module in slave mode. This means that the CNT_EN field is ignored and instead this PIT uses the master count enable signal broadcast from the PIT0 module. This bit allows synchronization of the counts across multiple PIT modules. This bit is only useful in designs with multiple PIT modules. Setting this bit in the (master) PIT0 module has no effect as its own CNT_EN field is also the master count enable. 0 CNT_EN from this PIT is used to control operation (default). 1 CNT_EN from master PIT is used to control operation.
14–10 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
9–8 CLKSEL	This field is used to select the source of the clocking for the counter. This field should not be changed when CNT_EN is set. The default selection is the IPBus clock. 00 Selects IPBus clock 01 Selects alternate clock 1 10 Selects alternate clock 2 11 Selects alternate clock 3
7 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
6–3 PRESCALER	This field is used to select the prescaling of the selected clock to determine the counting rate of the PIT. 0000 Clock 0001 Clock divided by 2 0010 Clock divided by 4 0011 Clock divided by 8 0100 Clock divided by 16 0101 Clock divided by 32 0110 Clock divided by 64 0111 Clock divided by 128 1000 Clock divided by 256 1001 Clock divided by 512 1010 Clock divided by 1024 1011 Clock divided by 2048 1100 Clock divided by 4096

Table continues on the next page...

PITx_CTRL field descriptions (continued)

Field	Description
	1101 Clock divided by 8192 1110 Clock divided by 16384 1111 Clock divided by 32768
2 PRF	PIT Roll-Over Flag. This bit is set when the counter rolls over to 0x0000 after matching the value in the PIT compare register. This bit is cleared by reading the CTRL register with PRF set and then writing a zero to this bit position. This bit can also be cleared by setting the CNT_EN bit to 0. Writing a one to the PRF bit position has no effect. 0 PIT counter has not reached the modulo value. (default) 1 PIT counter has reached the modulo value.
1 PRIE	PIT Roll-Over Interrupt Enable. This bit enables the PIT roll-over interrupt when the PRF bit becomes set. 0 PIT roll-over interrupt disabled (default). 1 PIT roll-over interrupt enabled.
0 CNT_EN	Count Enable This bit enables the PIT prescaler and counter. When this bit is clear, the counter remains at or returns to a 0x0000 value. The PRF bit is also reset when CNT_EN is clear. This field is ignored when the SLAVE bit is set and the count enable signal from the master PIT is used instead. 0 PIT counter reset (default). 1 PIT counter active.

31.2.2 PIT Modulo Register (PITx_MOD)

Address: Base address + 1h offset

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read Write	MODULO_VALUE[15:0]															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PITx_MOD field descriptions

Field	Description
MODULO_ VALUE[15:0]	This read/write register stores the modulo value for the PIT counter. When the PIT counter rolls over to 0x0000 from this value, the PRF bit becomes set and the PIT counter resumes counting from 0x0000.

31.2.3 PIT Counter Register (PITx_CNTR)

Address: Base address + 2h offset

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	COUNTER_VALUE															
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PITx_CNTR field descriptions

Field	Description
COUNTER_ VALUE	This read only register contains the current value of the PIT counter. Clearing CNT_EN resets the counter to 0x0000. When the PIT counter rolls over the modulo value, the PRF bit becomes set and the PIT counter resumes counting from 0x0000. If the selected counting rate is faster than the IPBus clock, then this count value cannot be read.

31.3 Functional Description

The purpose of the PIT is to create a repeated interrupt request at a programmable time interval. The periodic rate is determined based on the peripheral clock rate, the prescaler value, and the modulo value as shown in the following equation:

```
interrupt rate = peripheral clock rate / ((2^prescaler) * modulo value)
```

When using the PIT, set the clock select, prescaler, and modulo values prior to setting CNT_EN. Changing these settings with CNT_EN set may cause unexpected operation.

The roll-over flag is set upon rolling over the modulo value back to 0x0000. See the following figure for an example of PRF timing using a prescaler of 0x2 (IP bus clock divided by 4).

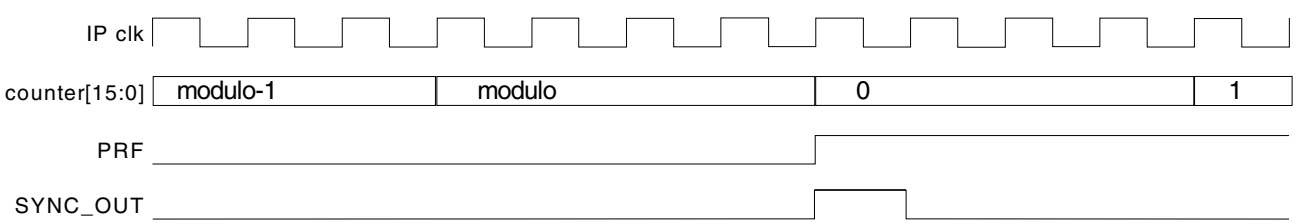


Figure 31-2. Example POF Timing

The roll-over flag can be cleared by writing a zero to the PRF bit position or by clearing CNT_EN.

31.3.1 Slave Mode

When using slave mode to synchronize several PIT modules only the CNT_EN signal is shared not the clocking for the counters. This means that each slave PIT must have their CLKSEL field, PRESCALER field, and PIT_MOD registers programmed prior to setting CNT_EN in the master PIT. The following figure shows the connection between the master PIT (PIT0) and the possible slave PITs (PIT1–PITn).

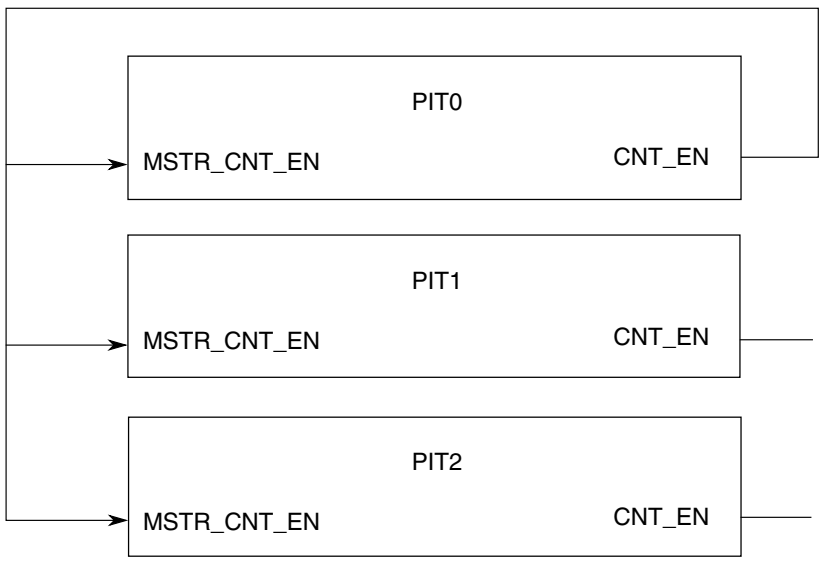


Figure 31-3. CNT_EN Connection Between Multiple PITs

31.3.2 Low Power Modes

This section describes the PIT low power modes.

31.3.2.1 Wait Mode

If the CNT_EN bit is set prior to entering wait mode, then the PIT continues to count and can wake the chip by asserting its interrupt upon reaching the modulo value.

31.3.2.2 Stop Mode

Stop mode operation depends on whether the system integration module (SIM) is set to allow the PIT to be clocked in stop mode. If not, the PIT counter does not operate during stop mode, but does retain its current settings. If CNT_EN is set, then the counter

resumes counting upon exit of stop mode assuming the exit isn't caused by a reset. If the PIT does receive clocks while the chip is in stop mode, then operation continues normally.

31.3.2.3 Debug Mode

If the CNT_EN bit is set prior to the chip entering debug mode, then the PIT continues to count during debug mode.

31.4 Interrupts

The PIT module can generate a single interrupt when the counter reaches the modulo value.

Chapter 32
CAN (FlexCAN)

32.1 Introduction

NOTE

For the chip-specific implementation details of this module's instances, see the chip configuration information.

The FlexCAN module is a communication controller implementing the CAN protocol according to the CAN 2.0 B protocol specification. A general block diagram is shown in the following figure, which describes the main subblocks implemented in the FlexCAN module, including one associated memory for storing message buffers, Receive (Rx) Global Mask registers, Receive Individual Mask registers, Receive FIFO filters, and Receive FIFO ID filters. The functions of the submodules are described in subsequent sections.

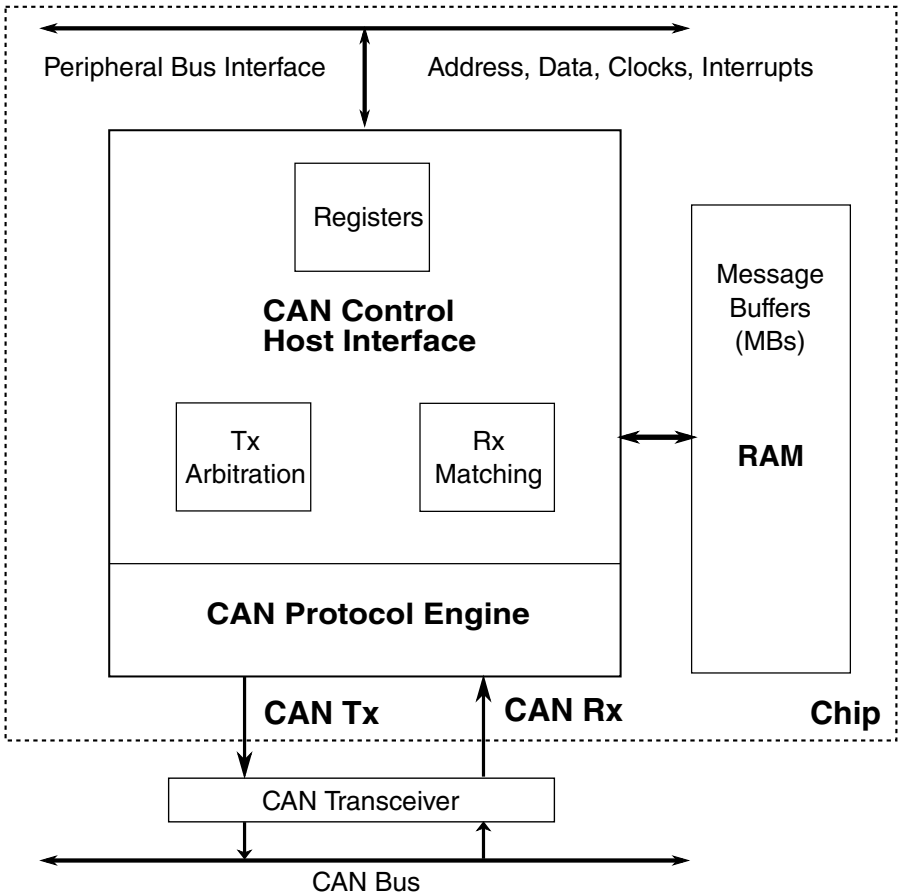


Figure 32-1. FlexCAN block diagram

32.1.1 Overview

The CAN protocol was primarily designed to be used as a vehicle serial data bus, meeting the specific requirements of this field:

- Real-time processing
- Reliable operation in the EMI environment of a vehicle
- Cost-effectiveness
- Required bandwidth

The FlexCAN module is a full implementation of the CAN protocol specification, Version 2.0 B, which supports both standard and extended message frames. The message buffers are stored in an embedded RAM dedicated to the FlexCAN module. See the chip configuration details for the actual number of message buffers configured in the MCU.

The CAN Protocol Engine (PE) submodule manages the serial communication on the CAN bus:

- Requesting RAM access for receiving and transmitting message frames

- Validating received messages
- Performing error handling

The Controller Host Interface (CHI) sub-module handles message buffer selection for reception and transmission, taking care of arbitration and ID matching algorithms.

The Bus Interface Unit (BIU) sub-module controls the access to and from the internal interface bus, in order to establish connection to the CPU and to other blocks. Clocks, address and data buses, interrupt outputs and test signals are accessed through the BIU.

32.1.2 FlexCAN module features

The FlexCAN module includes these distinctive legacy features:

- Full implementation of the CAN protocol specification, Version 2.0 B
  - Standard data and remote frames
  - Extended data and remote frames
  - Zero to eight bytes data length
  - Programmable bit rate up to 1 Mb/sec
  - Content-related addressing
- Compliant with the ISO 11898-1 standard
- Flexible mailboxes of zero to eight bytes data length
- Each mailbox configurable as receive or transmit, all supporting standard and extended messages
- Individual Rx Mask registers per mailbox
- Full-featured Rx FIFO with storage capacity for up to six frames and automatic internal pointer handling
- Transmission abort capability
- Programmable clock source to the CAN Protocol Interface, either bus clock or crystal oscillator
- Unused structures space can be used as general purpose RAM space
- Listen-Only mode capability
- Programmable Loop-Back mode supporting self-test operation

- Programmable transmission priority scheme: lowest ID, lowest buffer number, or highest priority
- Time stamp based on 16-bit free-running timer
- Global network time, synchronized by a specific message
- Maskable interrupts
- Independence from the transmission medium (an external transceiver is assumed)
- Short latency time due to an arbitration scheme for high-priority messages
- Low power modes, with programmable wake up on bus activity

New major features are also provided:

- Remote request frames may be handled automatically or by software
- CAN bit time settings and configuration bits can only be written in Freeze mode
- Tx mailbox status (Lowest priority buffer or empty buffer)
- Identifier Acceptance Filter Hit Indicator (IDHIT) register for received frames
- SYNCH bit available in Error in Status 1 register to inform that the module is synchronous with CAN bus
- CRC status for transmitted message
- Rx FIFO Global Mask register
- Selectable priority between mailboxes and Rx FIFO during matching process
- Powerful Rx FIFO ID filtering, capable of matching incoming IDs against either 128 extended, 256 standard, or 512 partial (8 bit) IDs, with up to 32 individual masking capability
- 100% backward compatibility with previous FlexCAN version

32.1.3 Modes of operation

The FlexCAN module has these functional modes:

- Normal mode (User or Supervisor):

In Normal mode, the module operates receiving and/or transmitting message frames, errors are handled normally, and all CAN Protocol functions are enabled. User and Supervisor Modes differ in the access to some restricted control registers.

- Freeze mode:

  Freeze mode is enabled when the FRZ bit in MCR is asserted. If enabled, Freeze mode is entered when MCR[HALT] is set or when Debug mode is requested at MCU level and MCR[FRZ_ACK] is asserted by the FlexCAN. In this mode, no transmission or reception of frames is done and synchronicity to the CAN bus is lost. See Freeze mode for more information.

- Listen-Only mode:

  The module enters this mode when the LOM field in the Control 1 Register is asserted. In this mode, transmission is disabled, all error counters are frozen, and the module operates in a CAN Error Passive mode. Only messages acknowledged by another CAN station will be received. If FlexCAN detects a message that has not been acknowledged, it will flag a BIT0 error (without changing the REC), as if it was trying to acknowledge the message.

- Loop-Back mode:

  The module enters this mode when the LPB field in the Control 1 Register is asserted. In this mode, FlexCAN performs an internal loop back that can be used for self-test operation. The bit stream output of the transmitter is internally fed back to the receiver input. The Rx CAN input pin is ignored and the Tx CAN output goes to the recessive state (logic '1'). FlexCAN behaves as it normally does when transmitting and treats its own transmitted message as a message received from a remote node. In this mode, FlexCAN 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.

For low-power operation, the FlexCAN module has:

- Module Disable mode:

  This low-power mode is entered when the MDIS bit in the MCR Register is asserted by the CPU and the LPM_ACK is asserted by the FlexCAN. When disabled, the module requests to disable the clocks to the CAN Protocol Engine and Controller Host Interface submodules. Exit from this mode is done by negating the MDIS bit in the MCR register. See Module Disable mode for more information.

- Doze mode:

This low power mode is entered when the DOZE bit in MCR is asserted and Doze mode is requested at MCU level and the LPM_ACK bit in the MCR Register is asserted by the FlexCAN. When in Doze mode, the module requests to disable the clocks to the CAN Protocol Engine and the CAN Controller-Host Interface sub-modules. Exit from this mode happens when the DOZE bit in MCR is negated, when the MCU is removed from Doze mode, or when activity is detected on the CAN bus and the Self Wake Up mechanism is enabled. See Doze mode for more information.

- Stop mode:

  This low power mode is entered when Stop mode is requested at MCU level and the LPM_ACK bit in the MCR Register is asserted by the FlexCAN. When in Stop Mode, the module puts itself in an inactive state and then informs the CPU that the clocks can be shut down globally. Exit from this mode happens when the Stop mode request is removed, or when activity is detected on the CAN bus and the Self Wake Up mechanism is enabled. See Stop mode for more information.

32.2 FlexCAN signal descriptions

The FlexCAN module has two I/O signals connected to the external MCU pins. These signals are summarized in the following table and described in more detail in the next subsections.

Table 32-1. FlexCAN signal descriptions

Signal	Description	I/O
CAN Rx	CAN Receive Pin	Input
CAN Tx	CAN Transmit Pin	Output

32.2.1 CAN Rx

This pin is the receive pin from the CAN bus transceiver. Dominant state is represented by logic level 0. Recessive state is represented by logic level 1.

32.2.2 CAN Tx

This pin is the transmit pin to the CAN bus transceiver. Dominant state is represented by logic level 0. Recessive state is represented by logic level 1.

32.3 Memory map/register definition

This section describes the registers and data structures in the FlexCAN module. The base address of the module depends on the particular memory map of the MCU.

32.3.1 FlexCAN memory mapping

The complete memory map for a FlexCAN module is shown in the following table.

The address space occupied by FlexCAN has 128 bytes for registers starting at the module base address, followed by embedded RAM starting at address 0x0080.

Each individual register is identified by its complete name and the corresponding mnemonic. The access type can be Supervisor (S) or Unrestricted (U). Most of the registers can be configured to have either Supervisor or Unrestricted access by programming the SUPV field in the MCR register. These registers are identified as S/U in the Access column of Table 32-2.

Table 32-2. Register access and reset information

Register	Access type	Affected by hard reset	Affected by soft reset
Module Configuration Register (MCR)	S	Yes	Yes
Control 1 register (CTRL1)	S/U	Yes	No
Free Running Timer register (TIMER)	S/U	Yes	Yes
Rx Mailboxes Global Mask register (RXMGMASK)	S/U	No	No
Rx Buffer 14 Mask register (RX14MASK)	S/U	No	No
Rx Buffer 15 Mask register (RX15MASK)	S/U	No	No
Error Counter Register (ECR)	S/U	Yes	Yes
Error and Status 1 Register (ESR1)	S/U	Yes	Yes
Interrupt Masks 1 register (IMASK1)	S/U	Yes	Yes
Interrupt Flags 1 register (IFLAG1)	S/U	Yes	Yes
Control 2 Register (CTRL2)	S/U	Yes	No
Error and Status 2 Register (ESR2)	S/U	Yes	Yes
CRC Register (CRCR)	S/U	Yes	Yes
Rx FIFO Global Mask register (RXFGMASK)	S/U	No	No
Rx FIFO Information Register (RXFIR)	S/U	No	No
Message buffers	S/U	No	No
Rx Individual Mask Registers	S/U	No	No

The FlexCAN module can store CAN messages for transmission and reception using mailboxes and Rx FIFO structures.

This module's memory map includes sixteen 128-bit message buffers (MBs) that occupy the range from offset 0x80 to 0x17F.

CAN memory map

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
E800	Module Configuration Register (CAN_MCR)	32	R/W	D890_000Fh	32.3.2/792
E802	Control 1 Register (CAN_CTRL1)	32	R/W	0000_0000h	32.3.3/797
E804	Free Running Timer (CAN_TIMER)	32	R/W	0000_0000h	32.3.4/800
E808	Rx Mailboxes Global Mask Register (CAN_RXMGMASK)	32	R/W	FFFF_FFFFh	32.3.5/801
E80A	Rx 14 Mask Register (CAN_RX14MASK)	32	R/W	FFFF_FFFFh	32.3.6/802
E80C	Rx 15 Mask Register (CAN_RX15MASK)	32	R/W	FFFF_FFFFh	32.3.7/802
E80E	Error Counter (CAN_ECR)	32	R/W	0000_0000h	32.3.8/803
E810	Error and Status 1 Register (CAN_ESR1)	32	R/W	0000_0000h	32.3.9/804
E812	Interrupt Masks 2 Register (CAN_IMASK2)	32	R/W	0000_0000h	32.3.10/ 809
E814	Interrupt Masks 1 Register (CAN_IMASK1)	32	R/W	0000_0000h	32.3.11/ 809
E816	Interrupt Flags 2 Register (CAN_IFLAG2)	32	R/W	0000_0000h	32.3.12/ 810
E818	Interrupt Flags 1 Register (CAN_IFLAG1)	32	R/W	0000_0000h	32.3.13/ 810
E81A	Control 2 Register (CAN_CTRL2)	32	R/W	00B0_0000h	32.3.14/ 813
E81C	Error and Status 2 Register (CAN_ESR2)	32	R/W	0000_0000h	32.3.15/ 816
E822	CRC Register (CAN_CRCR)	32	R	0000_0000h	32.3.16/ 817
E824	Rx FIFO Global Mask Register (CAN_RXFGMASK)	32	R/W	FFFF_FFFFh	32.3.17/ 818
E826	Rx FIFO Information Register (CAN_RXFIR)	32	R	Undefined	32.3.18/ 819
EC40	Rx Individual Mask Registers (CAN_RXIMR0)	32	R/W	Undefined	32.3.19/ 820
EC42	Rx Individual Mask Registers (CAN_RXIMR1)	32	R/W	Undefined	32.3.19/ 820
EC44	Rx Individual Mask Registers (CAN_RXIMR2)	32	R/W	Undefined	32.3.19/ 820
EC46	Rx Individual Mask Registers (CAN_RXIMR3)	32	R/W	Undefined	32.3.19/ 820

Table continues on the next page...

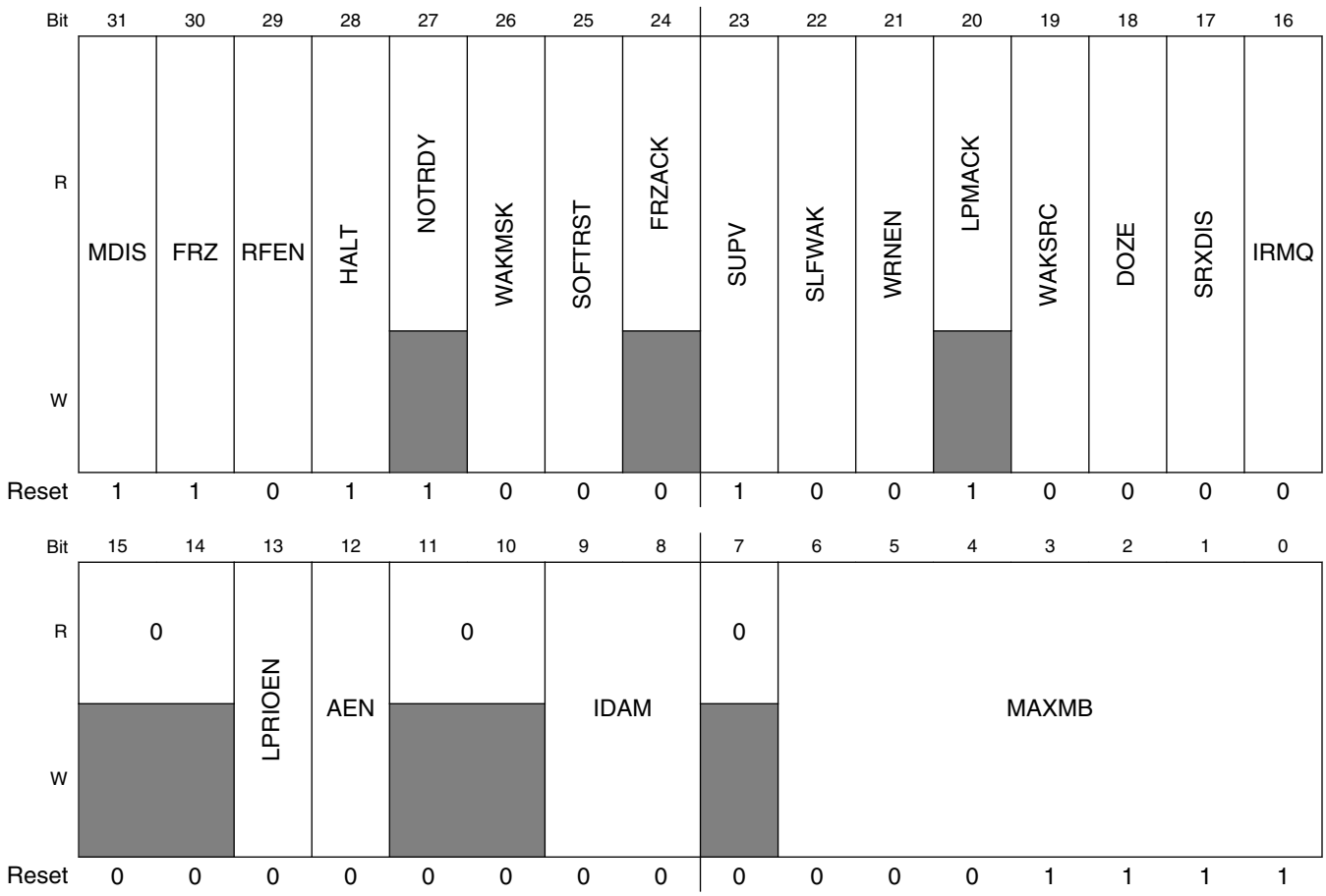
CAN memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
EC48	Rx Individual Mask Registers (CAN_RXIMR4)	32	R/W	Undefined	32.3.19/ 820
EC4A	Rx Individual Mask Registers (CAN_RXIMR5)	32	R/W	Undefined	32.3.19/ 820
EC4C	Rx Individual Mask Registers (CAN_RXIMR6)	32	R/W	Undefined	32.3.19/ 820
EC4E	Rx Individual Mask Registers (CAN_RXIMR7)	32	R/W	Undefined	32.3.19/ 820
EC50	Rx Individual Mask Registers (CAN_RXIMR8)	32	R/W	Undefined	32.3.19/ 820
EC52	Rx Individual Mask Registers (CAN_RXIMR9)	32	R/W	Undefined	32.3.19/ 820
EC54	Rx Individual Mask Registers (CAN_RXIMR10)	32	R/W	Undefined	32.3.19/ 820
EC56	Rx Individual Mask Registers (CAN_RXIMR11)	32	R/W	Undefined	32.3.19/ 820
EC58	Rx Individual Mask Registers (CAN_RXIMR12)	32	R/W	Undefined	32.3.19/ 820
EC5A	Rx Individual Mask Registers (CAN_RXIMR13)	32	R/W	Undefined	32.3.19/ 820
EC5C	Rx Individual Mask Registers (CAN_RXIMR14)	32	R/W	Undefined	32.3.19/ 820
EC5E	Rx Individual Mask Registers (CAN_RXIMR15)	32	R/W	Undefined	32.3.19/ 820

32.3.2 Module Configuration Register (CAN_MCR)

This register defines global system configurations, such as the module operation modes and the maximum message buffer configuration.

Address: E800h base + 0h offset = E800h



CAN_MCR field descriptions

Field	Description
31 MDIS	Module Disable This bit controls whether FlexCAN is enabled or not. When disabled, FlexCAN disables the clocks to the CAN Protocol Engine and Controller Host Interface sub-modules. This is the only bit in MCR not affected by soft reset. 0 Enable the FlexCAN module. 1 Disable the FlexCAN module.
30 FRZ	Freeze Enable The FRZ bit specifies the FlexCAN behavior when the HALT bit in the MCR Register is set or when Debug Mode is requested at MCU level. When FRZ is asserted, FlexCAN is enabled to enter Freeze Mode. Negation of this bit field causes FlexCAN to exit from Freeze Mode.

Table continues on the next page...

CAN_MCR field descriptions (continued)

Field	Description
	0 Not enabled to enter Freeze Mode 1 Enabled to enter Freeze Mode
29 RFEN	Rx FIFO Enable This bit controls whether the Rx FIFO feature is enabled or not. When RFEN is set, MBs 0 to 5 cannot be used for normal reception and transmission because the corresponding memory region (0x80-0xDC) is used by the FIFO engine as well as additional MBs (up to 32, depending on CTRL2[RFFN] setting) which are used as Rx FIFO ID Filter Table elements. RFEN also impacts the definition of the minimum number of peripheral clocks per CAN bit as described in the table "Minimum Ratio Between Peripheral Clock Frequency and CAN Bit Rate" (in section "Arbitration and Matching Timing"). This bit can only be written in Freeze mode as it is blocked by hardware in other modes. 0 Rx FIFO not enabled 1 Rx FIFO enabled
28 HALT	Halt FlexCAN Assertion of this bit puts the FlexCAN module into Freeze Mode. The CPU should clear it after initializing the Message Buffers and Control Register. No reception or transmission is performed by FlexCAN before this bit is cleared. Freeze Mode cannot be entered while FlexCAN is in a low power mode. 0 No Freeze Mode request. 1 Enters Freeze Mode if the FRZ bit is asserted.
27 NOTRDY	FlexCAN Not Ready This read-only bit indicates that FlexCAN is either in Disable Mode, Doze Mode, Stop Mode or Freeze Mode. It is negated once FlexCAN has exited these modes. 0 FlexCAN module is either in Normal Mode, Listen-Only Mode or Loop-Back Mode. 1 FlexCAN module is either in Disable Mode, Doze Mode, Stop Mode or Freeze Mode.
26 WAKMSK	Wake Up Interrupt Mask This bit enables the Wake Up Interrupt generation. 0 Wake Up Interrupt is disabled. 1 Wake Up Interrupt is enabled.
25 SOFTRST	Soft Reset When this bit is asserted, FlexCAN resets its internal state machines and some of the memory mapped registers. The following registers are reset: MCR (except the MDIS bit), TIMER, ECR, ESR1, ESR2, IMASK1, IMASK2, IFLAG1, IFLAG2 and CRCR. Configuration registers that control the interface to the CAN bus are not affected by soft reset. The following registers are unaffected: CTRL1, CTRL2, RXIMR0–RXIMR63, RXMGMASK, RX14MASK, RX15MASK, RXFGMASK, RXFIR, all Message Buffers. The SOFTRST bit can be asserted directly by the CPU when it writes to the MCR Register, but it is also asserted when global soft reset is requested at MCU level. Since soft reset is synchronous and has to follow a request/acknowledge procedure across clock domains, it may take some time to fully propagate its effect. The SOFTRST bit remains asserted while reset is pending, and is automatically negated when reset completes. Therefore, software can poll this bit to know when the soft reset has completed. Soft reset cannot be applied while clocks are shut down in a low power mode. The module should be first removed from low power mode, and then soft reset can be applied. 0 No reset request 1 Resets the registers affected by soft reset.

Table continues on the next page...

CAN_MCR field descriptions (continued)

Field	Description
24 FRZACK	Freeze Mode Acknowledge This read-only bit indicates that FlexCAN is in Freeze Mode and its prescaler is stopped. The Freeze Mode request cannot be granted until current transmission or reception processes have finished. Therefore the software can poll the FRZACK bit to know when FlexCAN has actually entered Freeze Mode. If Freeze Mode request is negated, then this bit is negated once the FlexCAN prescaler is running again. If Freeze Mode is requested while FlexCAN is in a low power mode, then the FRZACK bit will only be set when the low power mode is exited. See Section "Freeze Mode". **NOTE:** FRZACK will be asserted within 178 CAN bits from the freeze mode request by the CPU, and negated within 2 CAN bits after the freeze mode request removal (see Section "Protocol Timing"). 0 FlexCAN not in Freeze Mode, prescaler running 1 FlexCAN in Freeze Mode, prescaler stopped
23 SUPV	Supervisor Mode This bit configures the FlexCAN to be either in Supervisor or User Mode. The registers affected by this bit are marked as S/U in the Access Type column of the module memory map. Reset value of this bit is '1', so the affected registers start with Supervisor access allowance only. This bit can only be written in Freeze mode as it is blocked by hardware in other modes. 0 FlexCAN is in User Mode. Affected registers allow both Supervisor and Unrestricted accesses. 1 FlexCAN is in Supervisor Mode. Affected registers allow only Supervisor access. Unrestricted access behaves as though the access was done to an unimplemented register location.
22 SLFWAK	Self Wake Up This bit enables the Self Wake Up feature when FlexCAN is in a low power mode other than Disable Mode. When this feature is enabled, the FlexCAN module monitors the bus for wake up event, that is, a recessive-to-dominant transition. If a wake up event is detected during Doze Mode, FlexCAN requests to resume its clocks and, if enabled to do so, generates a Wake Up interrupt to the CPU. If a wake up event is detected during Stop Mode, then FlexCAN generates, if enabled to do so, a Wake Up interrupt to the CPU so that it can exit Stop Mode globally and FlexCAN can request to resume the clocks. When FlexCAN is in a low power mode other than Disable Mode, this bit cannot be written as it is blocked by hardware. 0 FlexCAN Self Wake Up feature is disabled. 1 FlexCAN Self Wake Up feature is enabled.
21 WRNEN	Warning Interrupt Enable When asserted, this bit enables the generation of the TWRNINT and RWRNINT flags in the Error and Status Register. If WRNEN is negated, the TWRNINT and RWRNINT flags will always be zero, independent of the values of the error counters, and no warning interrupt will ever be generated. This bit can only be written in Freeze mode as it is blocked by hardware in other modes. 0 TWRNINT and RWRNINT bits are zero, independent of the values in the error counters. 1 TWRNINT and RWRNINT bits are set when the respective error counter transitions from less than 96 to greater than or equal to 96.
20 LPMACK	Low Power Mode Acknowledge

Table continues on the next page...

CAN_MCR field descriptions (continued)

Field	Description
	This read-only bit indicates that FlexCAN is in a low power mode (Disable Mode, Doze Mode, Stop Mode). A low power mode can not be entered until all current transmission or reception processes have finished, so the CPU can poll the LPMACK bit to know when FlexCAN has actually entered low power mode. **NOTE:** LPMACK will be asserted within 180 CAN bits from the low power mode request by the CPU, and negated within 2 CAN bits after the low power mode request removal (see Section "Protocol Timing"). 0 FlexCAN is not in a low power mode. 1 FlexCAN is in a low power mode.
19 WAKSRC	Wake Up Source This bit defines whether the integrated low-pass filter is applied to protect the Rx CAN input from spurious wake up. This bit can only be written in Freeze mode as it is blocked by hardware in other modes. 0 FlexCAN uses the unfiltered Rx input to detect recessive to dominant edges on the CAN bus. 1 FlexCAN uses the filtered Rx input to detect recessive to dominant edges on the CAN bus.
18 DOZE	Doze Mode Enable This bit defines whether FlexCAN is allowed to enter low power mode when Doze Mode is requested at MCU level. This bit is automatically reset when FlexCAN wakes up from Doze Mode upon detecting activity on the CAN bus (self wake-up enabled). 0 FlexCAN is not enabled to enter low power mode when Doze Mode is requested. 1 FlexCAN is enabled to enter low power mode when Doze Mode is requested.
17 SRXDIS	Self Reception Disable This bit defines whether FlexCAN is allowed to receive frames transmitted by itself. If this bit is asserted, frames transmitted by the module will not be stored in any MB, regardless if the MB is programmed with an ID that matches the transmitted frame, and no interrupt flag or interrupt signal will be generated due to the frame reception. This bit can only be written in Freeze mode as it is blocked by hardware in other modes. 0 Self reception enabled 1 Self reception disabled
16 IRMQ	Individual Rx Masking and Queue Enable This bit indicates whether Rx matching process will be based either on individual masking and queue or on masking scheme with RXMGMASK, RX14MASK and RX15MASK, RXFGMASK. This bit can only be written in Freeze mode as it is blocked by hardware in other modes. 0 Individual Rx masking and queue feature are disabled. For backward compatibility, the reading of C/S word locks the MB even if it is EMPTY. 1 Individual Rx masking and queue feature are enabled.
15–14 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
13 LPRIOEN	Local Priority Enable This bit is provided for backwards compatibility reasons. It controls whether the local priority feature is enabled or not. It is used to expand the ID used during the arbitration process. With this expanded ID concept, the arbitration process is done based on the full 32-bit word, but the actual transmitted ID still has 11-bit for standard frames and 29-bit for extended frames. This bit can only be written in Freeze mode as it is blocked by hardware in other modes.

Table continues on the next page...

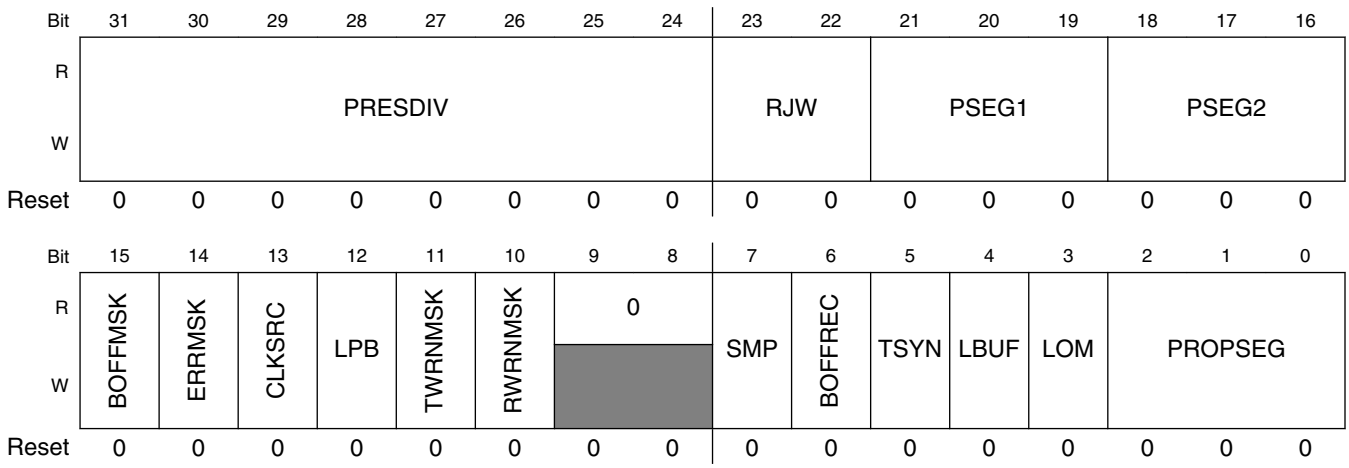
CAN_MCR field descriptions (continued)

Field	Description
	0 Local Priority disabled 1 Local Priority enabled
12 AEN	Abort Enable This bit is supplied for backwards compatibility reasons. When asserted, it enables the Tx abort mechanism. This mechanism guarantees a safe procedure for aborting a pending transmission, so that no frame is sent in the CAN bus without notification. This bit can only be written in Freeze mode as it is blocked by hardware in other modes. **NOTE:** When MCR[AEN] is asserted, only the abort mechanism (see Section "Transmission Abort Mechanism") must be used for updating Mailboxes configured for transmission. **CAUTION:** Writing the Abort code into Rx Mailboxes can cause unpredictable results when the MCR[AEN] is asserted. 0 Abort disabled 1 Abort enabled
11–10 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
9–8 IDAM	ID Acceptance Mode This 2-bit field identifies the format of the Rx FIFO ID Filter Table Elements. Note that all elements of the table are configured at the same time by this field (they are all the same format). See Section "Rx FIFO Structure". This field can only be written in Freeze mode as it is blocked by hardware in other modes. 00 Format A: One full ID (standard and extended) per ID Filter Table element. 01 Format B: Two full standard IDs or two partial 14-bit (standard and extended) IDs per ID Filter Table element. 10 Format C: Four partial 8-bit Standard IDs per ID Filter Table element. 11 Format D: All frames rejected.
7 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
MAXMB	Number of the Last Message Buffer This 7-bit field defines the number of the last Message Buffers that will take part in the matching and arbitration processes. The reset value (0x0F) is equivalent to 16 MB configuration. This field can only be written in Freeze Mode as it is blocked by hardware in other modes. Number of the last MB = MAXMB **NOTE:** MAXMB must be programmed with a value smaller than the parameter NUMBER_OF_MB, otherwise the number of the last effective Message Buffer will be: (NUMBER_OF_MB - 1) Additionally, the value of MAXMB must encompass the FIFO size defined by CTRL2[RFFN]. MAXMB also impacts the definition of the minimum number of peripheral clocks per CAN bit as described in Table "Minimum Ratio Between Peripheral Clock Frequency and CAN Bit Rate" (in Section "Arbitration and Matching Timing").

32.3.3 Control 1 Register (CAN_CTRL1)

This register is defined for specific FlexCAN control features related to the CAN bus, such as bit-rate, programmable sampling point within an Rx bit, Loop Back Mode, Listen-Only Mode, Bus Off recovery behavior and interrupt enabling (Bus-Off, Error, Warning). It also determines the Division Factor for the clock prescaler.

Address: E800h base + 2h offset = E802h



CAN_CTRL1 field descriptions

Field	Description
31–24 PRESDIV	Prescaler Division Factor This 8-bit field defines the ratio between the PE clock frequency and the Serial Clock (Sclock) frequency. The Sclock period defines the time quantum of the CAN protocol. For the reset value, the Sclock frequency is equal to the PE clock frequency. The Maximum value of this field is 0xFF, that gives a minimum Sclock frequency equal to the PE clock frequency divided by 256. See Section "Protocol Timing". This field can only be written in Freeze mode as it is blocked by hardware in other modes. Sclock frequency = PE clock frequency / (PRESDIV + 1)
23–22 RJW	Resync Jump Width This 2-bit field defines the maximum number of time quanta that a bit time can be changed by one re-synchronization. (One time quantum is equal to the Sclock period.) The valid programmable values are 0–3. This field can only be written in Freeze mode as it is blocked by hardware in other modes. Resync Jump Width = RJW + 1.
21–19 PSEG1	Phase Segment 1 This 3-bit field defines the length of Phase Buffer Segment 1 in the bit time. The valid programmable values are 0–7. This field can only be written in Freeze mode as it is blocked by hardware in other modes. Phase Buffer Segment 1 = (PSEG1 + 1) x Time-Quanta.
18–16 PSEG2	Phase Segment 2 This 3-bit field defines the length of Phase Buffer Segment 2 in the bit time. The valid programmable values are 1–7. This field can only be written in Freeze mode as it is blocked by hardware in other modes.

Table continues on the next page...

CAN_CTRL1 field descriptions (continued)

Field	Description
	Phase Buffer Segment 2 = (PSEG2 + 1) x Time-Quanta.
15 BOFFMSK	Bus Off Mask This bit provides a mask for the Bus Off Interrupt. 0 Bus Off interrupt disabled 1 Bus Off interrupt enabled
14 ERRMSK	Error Mask This bit provides a mask for the Error Interrupt. 0 Error interrupt disabled 1 Error interrupt enabled
13 CLKSRC	CAN Engine Clock Source This bit selects the clock source to the CAN Protocol Engine (PE) to be either the peripheral clock (driven by the PLL) or the crystal oscillator clock. The selected clock is the one fed to the prescaler to generate the Serial Clock (Sclock). In order to guarantee reliable operation, this bit can only be written in Disable mode as it is blocked by hardware in other modes. See Section "Protocol Timing". 0 The CAN engine clock source is the oscillator clock. Under this condition, the oscillator clock frequency must be lower than the bus clock. 1 The CAN engine clock source is the peripheral clock.
12 LPB	Loop Back Mode This bit configures FlexCAN to operate in Loop-Back Mode. In this mode, FlexCAN performs an internal loop back that can be used for self test operation. The bit stream output of the transmitter is fed back internally to the receiver input. The Rx CAN input pin is ignored and the Tx CAN output goes to the recessive state (logic '1'). FlexCAN behaves as it normally does when transmitting, and treats its own transmitted message as a message received from a remote node. In this mode, FlexCAN ignores the bit sent during the ACK slot in the CAN frame acknowledge field, generating an internal acknowledge bit to ensure proper reception of its own message. Both transmit and receive interrupts are generated. This bit can only be written in Freeze mode as it is blocked by hardware in other modes. **NOTE:** In this mode, the MCR[SRXDIS] cannot be asserted because this will impede the self reception of a transmitted message. 0 Loop Back disabled 1 Loop Back enabled
11 TWRNMSK	Tx Warning Interrupt Mask This bit provides a mask for the Tx Warning Interrupt associated with the TWRNINT flag in the Error and Status Register. This bit is read as zero when MCR[WRNEN] bit is negated. This bit can only be written if MCR[WRNEN] bit is asserted. 0 Tx Warning Interrupt disabled 1 Tx Warning Interrupt enabled
10 RWRNMSK	Rx Warning Interrupt Mask This bit provides a mask for the Rx Warning Interrupt associated with the RWRNINT flag in the Error and Status Register. This bit is read as zero when MCR[WRNEN] bit is negated. This bit can only be written if MCR[WRNEN] bit is asserted.

Table continues on the next page...

CAN_CTRL1 field descriptions (continued)

Field	Description
	0 Rx Warning Interrupt disabled 1 Rx Warning Interrupt enabled
9–8 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
7 SMP	CAN Bit Sampling This bit defines the sampling mode of CAN bits at the Rx input. This bit can only be written in Freeze mode as it is blocked by hardware in other modes. 0 Just one sample is used to determine the bit value. 1 Three samples are used to determine the value of the received bit: the regular one (sample point) and 2 preceding samples; a majority rule is used.
6 BOFFREC	Bus Off Recovery This bit defines how FlexCAN recovers from Bus Off state. If this bit is negated, automatic recovering from Bus Off state occurs according to the CAN Specification 2.0B. If the bit is asserted, automatic recovering from Bus Off is disabled and the module remains in Bus Off state until the bit is negated by the user. If the negation occurs before 128 sequences of 11 recessive bits are detected on the CAN bus, then Bus Off recovery happens as if the BOFFREC bit had never been asserted. If the negation occurs after 128 sequences of 11 recessive bits occurred, then FlexCAN will re-synchronize to the bus by waiting for 11 recessive bits before joining the bus. After negation, the BOFFREC bit can be re-asserted again during Bus Off, but it will only be effective the next time the module enters Bus Off. If BOFFREC was negated when the module entered Bus Off, asserting it during Bus Off will not be effective for the current Bus Off recovery. 0 Automatic recovering from Bus Off state enabled, according to CAN Spec 2.0 part B 1 Automatic recovering from Bus Off state disabled
5 TSYN	Timer Sync This bit enables a mechanism that resets the free-running timer each time a message is received in Message Buffer 0. This feature provides means to synchronize multiple FlexCAN stations with a special “SYNC” message (i.e., global network time). If the RFEN bit in MCR is set (Rx FIFO enabled), the first available Mailbox, according to CTRL2[RFFN] setting, is used for timer synchronization instead of MB0. This bit can only be written in Freeze mode as it is blocked by hardware in other modes. 0 Timer Sync feature disabled 1 Timer Sync feature enabled
4 LBUF	Lowest Buffer Transmitted First This bit defines the ordering mechanism for Message Buffer transmission. When asserted, the LPRIOEN bit does not affect the priority arbitration. This bit can only be written in Freeze mode as it is blocked by hardware in other modes. 0 Buffer with highest priority is transmitted first. 1 Lowest number buffer is transmitted first.
3 LOM	Listen-Only Mode This bit configures FlexCAN to operate in Listen-Only Mode. In this mode, transmission is disabled, all error counters are frozen and the module operates in a CAN Error Passive mode. Only messages acknowledged by another CAN station will be received. If FlexCAN detects a message that has not been acknowledged, it will flag a BIT0 error (without changing the REC), as if it was trying to acknowledge the message.

Table continues on the next page...

CAN_CTRL1 field descriptions (continued)

Field	Description
	Listen-Only Mode acknowledgement can be obtained by the state of ESR1[FLTCONF] field which is Passive Error when Listen-Only Mode is entered. There can be some delay between the Listen-Only Mode request and acknowledge. This bit can only be written in Freeze mode as it is blocked by hardware in other modes. 0 Listen-Only Mode is deactivated. 1 FlexCAN module operates in Listen-Only Mode.
PROPSEG	Propagation Segment This 3-bit field defines the length of the Propagation Segment in the bit time. The valid programmable values are 0–7. This field can only be written in Freeze mode as it is blocked by hardware in other modes. Propagation Segment Time = (PROPSEG + 1) * Time-Quanta. Time-Quantum = one Sclock period.

32.3.4 Free Running Timer (CAN_TIMER)

This register represents a 16-bit free running counter that can be read and written by the CPU. The timer starts from 0x0 after Reset, counts linearly to 0xFFFF, and wraps around.

The timer is clocked by the FlexCAN bit-clock (which defines the baud rate on the CAN bus). During a message transmission/reception, it increments by one for each bit that is received or transmitted. When there is no message on the bus, it counts using the previously programmed baud rate. The timer is not incremented during Disable, Doze, Stop and Freeze Modes.

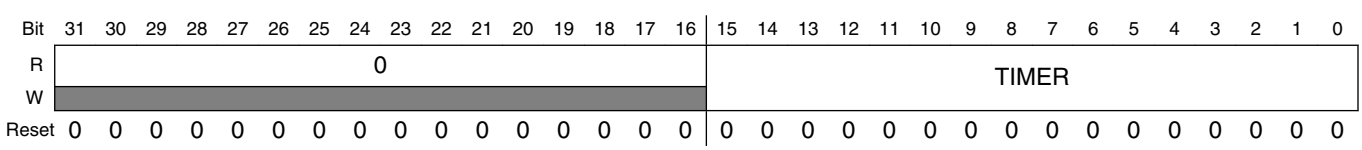
The timer value is captured when the second bit of the identifier field of any frame is on the CAN bus. This captured value is written into the Time Stamp entry in a message buffer after a successful reception or transmission of a message.

If bit CTRL1[TSYN] is asserted the Timer is reset whenever a message is received in the first available Mailbox, according to CTRL2[RFFN] setting.

The CPU can write to this register anytime. However, if the write occurs at the same time that the Timer is being reset by a reception in the first Mailbox, then the write value is discarded.

Reading this register affects the Mailbox Unlocking procedure; see Section "Message Buffer Lock Mechanism".

Address: E800h base + 4h offset = E804h



CAN_TIMER field descriptions

Field	Description
31–16 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
TIMER	Timer value Contains the free-running counter value.

32.3.5 Rx Mailboxes Global Mask Register (CAN_RXMGMASK)

This register is located in RAM.

RXMGMASK is provided for legacy support.

- When the MCR[IRMQ] bit is negated, RXMGMASK is always in effect.
- When the MCR[IRMQ] bit is asserted, RXMGMASK has no effect.

RXMGMASK is used to mask the filter fields of all Rx MBs, excluding MBs 14-15, which have individual mask registers.

This register can only be written in Freeze mode as it is blocked by hardware in other modes.

Address: E800h base + 8h offset = E808h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	MG[31:0]																															
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

CAN_RXMGMASK field descriptions

Field	Description
MG[31:0]	Rx Mailboxes Global Mask Bits These bits mask the Mailbox filter bits. Note that the alignment with the ID word of the Mailbox is not perfect as the two most significant MG bits affect the fields RTR and IDE, which are located in the Control and Status word of the Mailbox. The following table shows in detail which MG bits mask each Mailbox filter field.

SMB[RTR][1]	CTRL2[RRS]	CTRL2[EACEN]	Mailbox filter fields			
			MB[RTR]	MB[IDE]	MB[ID]	Reserved
0	-	0	note[2]	note[3]	MG[28:0]	MG[31:29]
0	-	1	MG[31]	MG[30]	MG[28:0]	MG[29]
1	0	-	-	-	-	MG[31:0]
1	1	0	-	-	MG[28:0]	MG[31:29]
1	1	1	MG[31]	MG[30]	MG[28:0]	MG[29]

CAN_RXMGMASK field descriptions (continued)

Field	Description
	1. RTR bit of the Incoming Frame. It is saved into an auxiliary MB called Rx Serial Message Buffer (Rx SMB). 2. If the CTRL2[EACEN] bit is negated, the RTR bit of Mailbox is never compared with the RTR bit of the incoming frame. 3. If the CTRL2[EACEN] bit is negated, the IDE bit of Mailbox is always compared with the IDE bit of the incoming frame. 0 The corresponding bit in the filter is "don't care." 1 The corresponding bit in the filter is checked.

1. RTR bit of the Incoming Frame. It is saved into an auxiliary MB called Rx Serial Message Buffer (Rx SMB).
2. If the CTRL2[EACEN] bit is negated, the RTR bit of Mailbox is never compared with the RTR bit of the incoming frame.
3. If the CTRL2[EACEN] bit is negated, the IDE bit of Mailbox is always compared with the IDE bit of the incoming frame.

32.3.6 Rx 14 Mask Register (CAN_RX14MASK)

This register is located in RAM.

RX14MASK is provided for legacy support. When the MCR[IRMQ] bit is asserted, RX14MASK has no effect.

RX14MASK is used to mask the filter fields of Message Buffer 14.

This register can only be programmed while the module is in Freeze Mode as it is blocked by hardware in other modes.

Address: E800h base + Ah offset = E80Ah

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	RX14M[31:0]																															
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

CAN_RX14MASK field descriptions

Field	Description
RX14M[31:0]	Rx Buffer 14 Mask Bits Each mask bit masks the corresponding Mailbox 14 filter field in the same way that RXMGMASK masks other Mailboxes' filters. See the description of the CAN_RXMGMASK register. 0 The corresponding bit in the filter is "don't care." 1 The corresponding bit in the filter is checked.

32.3.7 Rx 15 Mask Register (CAN_RX15MASK)

This register is located in RAM.

RX15MASK is provided for legacy support. When the MCR[IRMQ] bit is asserted, RX15MASK has no effect.

RX15MASK is used to mask the filter fields of Message Buffer 15.

This register can only be programmed while the module is in Freeze Mode as it is blocked by hardware in other modes.

Address: E800h base + Ch offset = E80Ch

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	RX15M[31:0]																															
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

CAN_RX15MASK field descriptions

Field	Description
RX15M[31:0]	Rx Buffer 15 Mask Bits Each mask bit masks the corresponding Mailbox 15 filter field in the same way that RXMGMASK masks other Mailboxes' filters. See the description of the CAN_RXMGMASK register. 0 The corresponding bit in the filter is "don't care." 1 The corresponding bit in the filter is checked.

32.3.8 Error Counter (CAN_ECR)

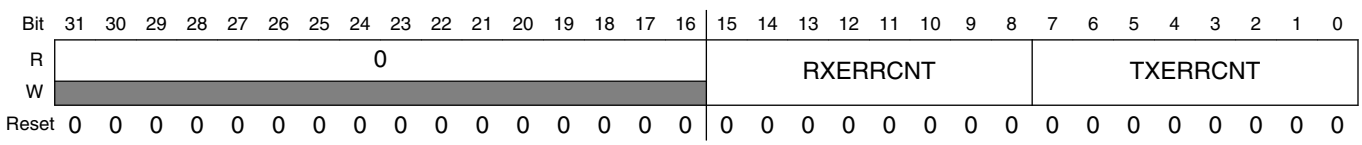
This register has two 8-bit fields reflecting the value of two FlexCAN error counters: Transmit Error Counter (TXERRCNT field) and Receive Error Counter (RXERRCNT field). The rules for increasing and decreasing these counters are described in the CAN protocol and are completely implemented in the FlexCAN module. Both counters are read-only except in Freeze Mode, where they can be written by the CPU.

FlexCAN responds to any bus state as described in the protocol, e.g. transmit 'Error Active' or 'Error Passive' flag, delay its transmission start time ('Error Passive') and avoid any influence on the bus when in 'Bus Off' state. The following are the basic rules for FlexCAN bus state transitions.

- If the value of TXERRCNT or RXERRCNT increases to be greater than or equal to 128, the FLTCONF field in the Error and Status Register is updated to reflect 'Error Passive' state.
- If the FlexCAN state is 'Error Passive', and either TXERRCNT or RXERRCNT decrements to a value less than or equal to 127 while the other already satisfies this condition, the FLTCONF field in the Error and Status Register is updated to reflect 'Error Active' state.

- If the value of TXERRCNT increases to be greater than 255, the FLTCONF field in the Error and Status Register is updated to reflect ‘Bus Off’ state, and an interrupt may be issued. The value of TXERRCNT is then reset to zero.
- If FlexCAN is in ‘Bus Off’ state, then TXERRCNT is cascaded together with another internal counter to count the 128th occurrences of 11 consecutive recessive bits on the bus. Hence, TXERRCNT is reset to zero and counts in a manner where the internal counter counts 11 such bits and then wraps around while incrementing the TXERRCNT. When TXERRCNT reaches the value of 128, the FLTCONF field in the Error and Status Register is updated to be ‘Error Active’ and both error counters are reset to zero. At any instance of dominant bit following a stream of less than 11 consecutive recessive bits, the internal counter resets itself to zero without affecting the TXERRCNT value.
- If during system start-up, only one node is operating, then its TXERRCNT increases in each message it is trying to transmit, as a result of acknowledge errors (indicated by the ACKERR bit in the Error and Status Register). After the transition to ‘Error Passive’ state, the TXERRCNT does not increment anymore by acknowledge errors. Therefore the device never goes to the ‘Bus Off’ state.
- If the RXERRCNT increases to a value greater than 127, it is not incremented further, even if more errors are detected while being a receiver. At the next successful message reception, the counter is set to a value between 119 and 127 to resume to ‘Error Active’ state.

Address: E800h base + Eh offset = E80Eh



CAN_ECR field descriptions

Field	Description
31–16 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
15–8 RXERRCNT	Receive Error Counter
TXERRCNT	Transmit Error Counter

32.3.9 Error and Status 1 Register (CAN_ESR1)

This register reflects various error conditions, some general status of the device and it is the source of interrupts to the CPU.

The CPU read action clears bits 15-10, therefore the reported error conditions (bits 15-10) are those that occurred since the last time the CPU read this register. Bits 9-3 are status bits.

The following table shows the FlexCAN state variables and their meanings. Other combinations not shown in the table are reserved.

SYNCH	IDLE	TX	RX	FlexCAN State
0	0	0	0	Not synchronized to CAN bus
1	1	x	x	Idle
1	0	1	0	Transmitting
1	0	0	1	Receiving

Address: E800h base + 10h offset = E810h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0													SYNCH	TWRNINT	RWRNINT
W															w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	BIT1ERR	BIT0ERR	ACKERR	CRCERR	FRMERR	STFERR	TXWRN	RXWRN	IDLE	TX	FLTCONF		RX	BOFFINT	ERRINT	WAKINT
W														w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

CAN_ESR1 field descriptions

Field	Description
31–19 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
18 SYNCH	CAN Synchronization Status This read-only flag indicates whether the FlexCAN is synchronized to the CAN bus and able to participate in the communication process. It is set and cleared by the FlexCAN. See the table in the overall CAN_ESR1 register description. 0 FlexCAN is not synchronized to the CAN bus. 1 FlexCAN is synchronized to the CAN bus.
17 TWRNINT	Tx Warning Interrupt Flag If the WRNEN bit in MCR is asserted, the TWRNINT bit is set when the TXWRN flag transitions from ‘0’ to ‘1’, meaning that the Tx error counter reached 96. If the corresponding mask bit in the Control Register (TWRNMSK) is set, an interrupt is generated to the CPU. This bit is cleared by writing it to ‘1’. When WRNEN is negated, this flag is masked. CPU must clear this flag before disabling the bit. Otherwise it will be set when the WRNEN is set again. Writing ‘0’ has no effect. This flag is not generated during “Bus Off” state. This bit is not updated during Freeze mode. 0 No such occurrence 1 The Tx error counter transitioned from less than 96 to greater than or equal to 96.
16 RWRNINT	Rx Warning Interrupt Flag If the WRNEN bit in MCR is asserted, the RWRNINT bit is set when the RXWRN flag transitions from ‘0’ to ‘1’, meaning that the Rx error counters reached 96. If the corresponding mask bit in the Control Register (RWRNMSK) is set, an interrupt is generated to the CPU. This bit is cleared by writing it to ‘1’. When WRNEN is negated, this flag is masked. CPU must clear this flag before disabling the bit. Otherwise it will be set when the WRNEN is set again. Writing ‘0’ has no effect. This bit is not updated during Freeze mode. 0 No such occurrence 1 The Rx error counter transitioned from less than 96 to greater than or equal to 96.
15 BIT1ERR	Bit1 Error This bit indicates when an inconsistency occurs between the transmitted and the received bit in a message. **NOTE:** This bit is not set by a transmitter in case of arbitration field or ACK slot, or in case of a node sending a passive error flag that detects dominant bits. 0 No such occurrence 1 At least one bit sent as recessive is received as dominant.
14 BIT0ERR	Bit0 Error This bit indicates when an inconsistency occurs between the transmitted and the received bit in a message. 0 No such occurrence 1 At least one bit sent as dominant is received as recessive.
13 ACKERR	Acknowledge Error This bit indicates that an Acknowledge Error has been detected by the transmitter node, i.e., a dominant bit has not been detected during the ACK SLOT.

Table continues on the next page...

CAN_ESR1 field descriptions (continued)

Field	Description
	0 No such occurrence 1 An ACK error occurred since last read of this register.
12 CRCERR	Cyclic Redundancy Check Error This bit indicates that a CRC Error has been detected by the receiver node, i.e., the calculated CRC is different from the received. 0 No such occurrence 1 A CRC error occurred since last read of this register.
11 FRMERR	Form Error This bit indicates that a Form Error has been detected by the receiver node, i.e., a fixed-form bit field contains at least one illegal bit. 0 No such occurrence 1 A Form Error occurred since last read of this register.
10 STFERR	Stuffing Error This bit indicates that a Stuffing Error has been detected. 0 No such occurrence 1 A Stuffing Error occurred since last read of this register.
9 TXWRN	TX Error Warning This bit indicates when repetitive errors are occurring during message transmission. This bit is not updated during Freeze mode. 0 No such occurrence 1 TXERRCNT is greater than or equal to 96.
8 RXWRN	Rx Error Warning This bit indicates when repetitive errors are occurring during message reception. This bit is not updated during Freeze mode. 0 No such occurrence 1 RXERRCNT is greater than or equal to 96.
7 IDLE	This bit indicates when CAN bus is in IDLE state. See the table in the overall CAN_ESR1 register description. 0 No such occurrence 1 CAN bus is now IDLE.
6 TX	FlexCAN in Transmission This bit indicates if FlexCAN is transmitting a message. See the table in the overall CAN_ESR1 register description. 0 FlexCAN is not transmitting a message. 1 FlexCAN is transmitting a message.
5–4 FLTCONF	Fault Confinement State This 2-bit field indicates the Confinement State of the FlexCAN module.

Table continues on the next page...

CAN_ESR1 field descriptions (continued)

Field	Description
	If the LOM bit in the Control Register is asserted, after some delay that depends on the CAN bit timing the FLTCONF field will indicate “Error Passive”. The very same delay affects the way how FLTCONF reflects an update to ECR register by the CPU. It may be necessary up to one CAN bit time to get them coherent again. Since the Control Register is not affected by soft reset, the FLTCONF field will not be affected by soft reset if the LOM bit is asserted. 00 Error Active 01 Error Passive 1x Bus Off
3 RX	FlexCAN in Reception This bit indicates if FlexCAN is receiving a message. See the table in the overall CAN_ESR1 register description. 0 FlexCAN is not receiving a message. 1 FlexCAN is receiving a message.
2 BOFFINT	‘Bus Off’ Interrupt This bit is set when FlexCAN enters ‘Bus Off’ state. If the corresponding mask bit in the Control Register (BOFFMSK) is set, an interrupt is generated to the CPU. This bit is cleared by writing it to ‘1’. Writing ‘0’ has no effect. 0 No such occurrence 1 FlexCAN module entered ‘Bus Off’ state
1 ERRINT	Error Interrupt This bit indicates that at least one of the Error Bits (bits 15-10) is set. If the corresponding mask bit CTRL1[ERRMSK] is set, an interrupt is generated to the CPU. This bit is cleared by writing it to ‘1’. Writing ‘0’ has no effect. 0 No such occurrence 1 Indicates setting of any Error Bit in the Error and Status Register
0 WAKINT	Wake-Up Interrupt This field applies when FlexCAN is in low power mode: • Doze Mode • Stop Mode When a recessive-to-dominant transition is detected on the CAN bus and if the MCR[WAKMSK] bit is set, an interrupt is generated to the CPU. This bit is cleared by writing it to ‘1’. When MCR[SLFWAK] is negated, this flag is masked. The CPU must clear this flag before disabling the bit. Otherwise it will be set when the SLFWAK is set again. Writing ‘0’ has no effect. 0 No such occurrence 1 Indicates a recessive to dominant transition was received on the CAN bus

32.3.10 Interrupt Masks 2 Register (CAN_IMASK2)

This register allows any number of a range of 32 Message Buffer Interrupts to be enabled or disabled. It contains one interrupt mask bit per buffer, enabling the CPU to determine which buffer generates an interrupt after a successful transmission or reception (i.e. when the corresponding IFLAG2 bit is set).

Address: E800h base + 12h offset = E812h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	BUFHM																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

CAN_IMASK2 field descriptions

Field	Description
BUFHM	Buffer MB_i Mask Each bit enables or disables the corresponding FlexCAN Message Buffer Interrupt. **NOTE:** Setting or clearing a bit in the IMASK2 Register can assert or negate an interrupt request, if the corresponding IFLAG2 bit is set. 0 The corresponding buffer Interrupt is disabled. 1 The corresponding buffer Interrupt is enabled.

32.3.11 Interrupt Masks 1 Register (CAN_IMASK1)

This register allows any number of a range of 32 Message Buffer Interrupts to be enabled or disabled. It contains one interrupt mask bit per buffer, enabling the CPU to determine which buffer generates an interrupt after a successful transmission or reception (i.e. when the corresponding IFLAG1 bit is set).

Address: E800h base + 14h offset = E814h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	BUFLM																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

CAN_IMASK1 field descriptions

Field	Description
BUFLM	Buffer MB_i Mask

CAN_IMASK1 field descriptions (continued)

Field	Description
	Each bit enables or disables the corresponding FlexCAN Message Buffer Interrupt. **NOTE:** Setting or clearing a bit in the IMASK1 Register can assert or negate an interrupt request, if the corresponding IFLAG1 bit is set. 0 The corresponding buffer Interrupt is disabled. 1 The corresponding buffer Interrupt is enabled.

32.3.12 Interrupt Flags 2 Register (CAN_IFLAG2)

This register defines the flags for 32 Message Buffer interrupts. It contains one interrupt flag bit per buffer. Each successful transmission or reception sets the corresponding IFLAG2 bit. If the corresponding IMASK2 bit is set, an interrupt will be generated. The interrupt flag must be cleared by writing 1 to it. Writing 0 has no effect.

Before updating MCR[MAXMB] field, CPU must service the IFLAG2 bits whose MB value is greater than the MCR[MAXMB] to be updated; otherwise, they will remain set and be inconsistent with the amount of MBs available.

Address: E800h base + 16h offset = E816h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	BUFHI																															
W	w1c																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

CAN_IFLAG2 field descriptions

Field	Description
BUFHI	Buffer MB_i Interrupt Each bit flags the corresponding FlexCAN Message Buffer interrupt. 0 The corresponding buffer has no occurrence of successfully completed transmission or reception. 1 The corresponding buffer has successfully completed transmission or reception.

32.3.13 Interrupt Flags 1 Register (CAN_IFLAG1)

This register defines the flags for 32 Message Buffer interrupts. It contains one interrupt flag bit per buffer. Each successful transmission or reception sets the corresponding IFLAG1 bit. If the corresponding IMASK1 bit is set, an interrupt will be generated. The interrupt flag must be cleared by writing 1 to it. Writing 0 has no effect.

The BUF7I to BUF5I flags are also used to represent FIFO interrupts when the Rx FIFO is enabled. When the bit MCR[RFEN] is set the function of the 8 least significant interrupt flags BUF[7:0]I changes: BUF7I, BUF6I and BUF5I indicate operating conditions of the FIFO, and BUF4TO0I are reserved.

Before enabling the RFEN, the CPU must service the IFLAG bits asserted in the Rx FIFO region; see Section "Rx FIFO". Otherwise, these IFLAG bits will mistakenly show the related MBs now belonging to FIFO as having contents to be serviced. When the RFEN bit is negated, the FIFO flags must be cleared. The same care must be taken when an RFFN value is selected extending Rx FIFO filters beyond MB7. For example, when RFFN is 0x8, the MB0-23 range is occupied by Rx FIFO filters and related IFLAG bits must be cleared.

Before updating MCR[MAXMB] field, CPU must service the IFLAG1 bits whose MB value is greater than the MCR[MAXMB] to be updated; otherwise, they will remain set and be inconsistent with the amount of MBs available.

Address: E800h base + 18h offset = E818h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	BUF31TO8I															
W	w1c															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	BUF31TO8I								BUF7I	BUF6I	BUF5I	BUF4TO0I				
W	w1c								w1c	w1c	w1c	w1c				
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

CAN_IFLAG1 field descriptions

Field	Description
31–8 BUF31TO8I	Buffer MB_i Interrupt Each bit flags the corresponding FlexCAN Message Buffer interrupt. 0 The corresponding buffer has no occurrence of successfully completed transmission or reception. 1 The corresponding buffer has successfully completed transmission or reception.

Table continues on the next page...

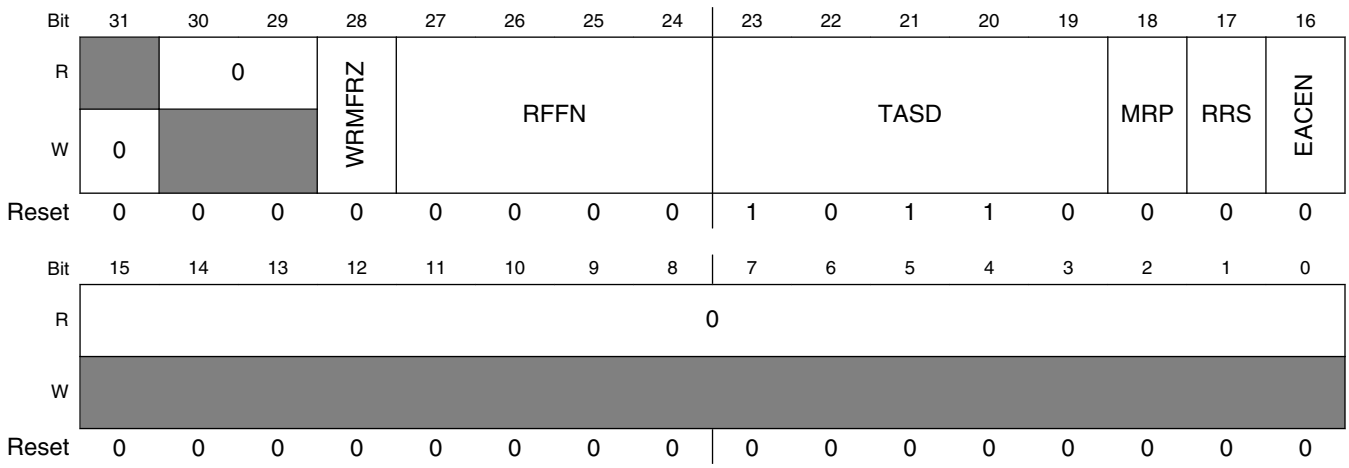
CAN_IFLAG1 field descriptions (continued)

Field	Description
7 BUF7I	Buffer MB7 Interrupt or "Rx FIFO Overflow" When the RFEN bit in the MCR is cleared (Rx FIFO disabled), this bit flags the interrupt for MB7. **NOTE:** This flag is cleared by the FlexCAN whenever the bit MCR[RFEN] is changed by CPU writes. The BUF7I flag represents "Rx FIFO Overflow" when MCR[RFEN] is set. In this case, the flag indicates that a message was lost because the Rx FIFO is full. Note that the flag will not be asserted when the Rx FIFO is full and the message was captured by a Mailbox. 0 No occurrence of MB7 completing transmission/reception (when MCR[RFEN]=0) or of Rx FIFO overflow (when MCR[RFEN]=1) 1 MB7 completed transmission/reception (when MCR[RFEN]=0) or Rx FIFO overflow (when MCR[RFEN]=1)
6 BUF6I	Buffer MB6 Interrupt or "Rx FIFO Warning" When the RFEN bit in the MCR is cleared (Rx FIFO disabled), this bit flags the interrupt for MB6. **NOTE:** This flag is cleared by the FlexCAN whenever the bit MCR[RFEN] is changed by CPU writes. The BUF6I flag represents "Rx FIFO Warning" when MCR[RFEN] is set. In this case, the flag indicates when the number of unread messages within the Rx FIFO is increased to 5 from 4 due to the reception of a new one, meaning that the Rx FIFO is almost full. Note that if the flag is cleared while the number of unread messages is greater than 4, it does not assert again until the number of unread messages within the Rx FIFO is decreased to be equal to or less than 4. 0 No occurrence of MB6 completing transmission/reception (when MCR[RFEN]=0) or of Rx FIFO almost full (when MCR[RFEN]=1) 1 MB6 completed transmission/reception (when MCR[RFEN]=0) or Rx FIFO almost full (when MCR[RFEN]=1)
5 BUF5I	Buffer MB5 Interrupt or "Frames available in Rx FIFO" When the RFEN bit in the MCR is cleared (Rx FIFO disabled), this bit flags the interrupt for MB5. **NOTE:** This flag is cleared by the FlexCAN whenever the bit MCR[RFEN] is changed by CPU writes. The BUF5I flag represents "Frames available in Rx FIFO" when MCR[RFEN] is set. In this case, the flag indicates that at least one frame is available to be read from the Rx FIFO. 0 No occurrence of MB5 completing transmission/reception (when MCR[RFEN]=0) or of frame(s) available in the Rx FIFO (when MCR[RFEN]=1) 1 MB5 completed transmission/reception (when MCR[RFEN]=0) or frame(s) available in the Rx FIFO (when MCR[RFEN]=1)
BUF4TO0I	Buffer MB_i Interrupt or "reserved" When the RFEN bit in the MCR is cleared (Rx FIFO disabled), these bits flag the interrupts for MB4 to MB0. **NOTE:** These flags are cleared by the FlexCAN whenever the bit MCR[RFEN] is changed by CPU writes. The BUF4TO0I flags are reserved when MCR[RFEN] is set. 0 The corresponding buffer has no occurrence of successfully completed transmission or reception (when MCR[RFEN]=0). 1 The corresponding buffer has successfully completed transmission or reception (when MCR[RFEN]=0).

32.3.14 Control 2 Register (CAN_CTRL2)

This register contains control bits for CAN errors, FIFO features, and mode selection.

Address: E800h base + 1Ah offset = E81Ah



CAN_CTRL2 field descriptions

Field	Description
31 Reserved	This field is reserved.
30–29 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
28 WRMFRZ	Write-Access to Memory in Freeze mode Enable unrestricted write access to FlexCAN memory in Freeze mode. This bit can only be written in Freeze mode and has no effect out of Freeze mode. 0 Maintain the write access restrictions. 1 Enable unrestricted write access to FlexCAN memory.
27–24 RFFN	Number of Rx FIFO Filters This 4-bit field defines the number of Rx FIFO filters, as shown in the following table. The maximum selectable number of filters is determined by the MCU. This field can only be written in Freeze mode as it is blocked by hardware in other modes. This field must not be programmed with values that make the number of Message Buffers occupied by Rx FIFO and ID Filter exceed the number of Mailboxes present, defined by MCR[MAXMB]. **NOTE:** Each group of eight filters occupies a memory space equivalent to two Message Buffers which means that the more filters are implemented the less Mailboxes will be available. Considering that the Rx FIFO occupies the memory space originally reserved for MB0-5, RFFN should be programmed with a value correponding to a number of filters not greater than the number of available memory words which can be calculated as follows: (SETUP_MB - 6) x 4 where SETUP_MB is the least between NUMBER_OF_MB and MAXMB. The number of remaining Mailboxes available will be:

Table continues on the next page...

CAN_CTRL2 field descriptions (continued)

<table>
<tr><th>Field</th><th>Description</th></tr>
<tr><td></td><td>
(SETUP_MB - 8) - (RFFN x 2)

If the Number of Rx FIFO Filters programmed through RFFN exceeds the SETUP_MB value (memory space available) the exceeding ones will not be functional.
<table>
<tr><th>RFFN[3:0]</th><th>Number of Rx FIFO filters</th><th>Message Buffers occupied by Rx FIFO and ID Filter Table</th><th>Remaining Available Mailboxes[1]</th><th>Rx FIFO ID Filter Table Elements Affected by Rx Individual Masks</th><th>Rx FIFO ID Filter Table Elements Affected by Rx FIFO Global Mask[2]</th></tr>
<tr><td>0x0</td><td>8</td><td>MB 0-7</td><td>MB 8-63</td><td>Elements 0-7</td><td>none</td></tr>
<tr><td>0x1</td><td>16</td><td>MB 0-9</td><td>MB 10-63</td><td>Elements 0-9</td><td>Elements 10-15</td></tr>
<tr><td>0x2</td><td>24</td><td>MB 0-11</td><td>MB 12-63</td><td>Elements 0-11</td><td>Elements 12-23</td></tr>
<tr><td>0x3</td><td>32</td><td>MB 0-13</td><td>MB 14-63</td><td>Elements 0-13</td><td>Elements 14-31</td></tr>
<tr><td>0x4</td><td>40</td><td>MB 0-15</td><td>MB 16-63</td><td>Elements 0-15</td><td>Elements 16-39</td></tr>
<tr><td>0x5</td><td>48</td><td>MB 0-17</td><td>MB 18-63</td><td>Elements 0-17</td><td>Elements 18-47</td></tr>
<tr><td>0x6</td><td>56</td><td>MB 0-19</td><td>MB 20-63</td><td>Elements 0-19</td><td>Elements 20-55</td></tr>
<tr><td>0x7</td><td>64</td><td>MB 0-21</td><td>MB 22-63</td><td>Elements 0-21</td><td>Elements 22-63</td></tr>
<tr><td>0x8</td><td>72</td><td>MB 0-23</td><td>MB 24-63</td><td>Elements 0-23</td><td>Elements 24-71</td></tr>
<tr><td>0x9</td><td>80</td><td>MB 0-25</td><td>MB 26-63</td><td>Elements 0-25</td><td>Elements 26-79</td></tr>
<tr><td>0xA</td><td>88</td><td>MB 0-27</td><td>MB 28-63</td><td>Elements 0-27</td><td>Elements 28-87</td></tr>
<tr><td>0xB</td><td>96</td><td>MB 0-29</td><td>MB 30-63</td><td>Elements 0-29</td><td>Elements 30-95</td></tr>
<tr><td>0xC</td><td>104</td><td>MB 0-31</td><td>MB 32-63</td><td>Elements 0-31</td><td>Elements 32-103</td></tr>
<tr><td>0xD</td><td>112</td><td>MB 0-33</td><td>MB 34-63</td><td>Elements 0-31</td><td>Elements 32-111</td></tr>
<tr><td>0xE</td><td>120</td><td>MB 0-35</td><td>MB 36-63</td><td>Elements 0-31</td><td>Elements 32-119</td></tr>
<tr><td>0xF</td><td>128</td><td>MB 0-37</td><td>MB 38-63</td><td>Elements 0-31</td><td>Elements 32-127</td></tr>
</table>
1. The number of the last remaining available mailboxes is defined by the least value between the parameter NUMBER_OF_MB minus 1 and the MCR[MAXMB] field.
2. If Rx Individual Mask Registers are not enabled then all Rx FIFO filters are affected by the Rx FIFO Global Mask.
</td></tr>
<tr><td>23–19
TASD</td><td>
Tx Arbitration Start Delay
This 5-bit field indicates how many CAN bits the Tx arbitration process start point can be delayed from the first bit of CRC field on CAN bus. This field can only be written in Freeze mode as it is blocked by hardware in other modes.

This field is useful to optimize the transmit performance based on factors such as: peripheral/serial clock ratio, CAN bit timing and number of MBs. The duration of an arbitration process, in terms of CAN bits, is directly proportional to the number of available MBs and CAN baud rate and inversely proportional to the peripheral clock frequency.

The optimal arbitration timing is that in which the last MB is scanned right before the first bit of the Intermission field of a CAN frame. Therefore, if there are few MBs and the system/serial clock ratio is high and the CAN baud rate is low then the arbitration can be delayed and vice-versa.

If TASD is 0 then the arbitration start is not delayed, thus the CPU has less time to configure a Tx MB for the next arbitration, but more time is reserved for arbitration. In the other hand, if TASD is 24 then the CPU can configure a Tx MB later and less time is reserved for arbitration.
</td></tr>
</table>

Table continues on the next page...

CAN_CTRL2 field descriptions (continued)

Field	Description
	If too little time is reserved for arbitration the FlexCAN may be not able to find winner MBs in time to compete with other nodes for the CAN bus. If the arbitration ends too much time before the first bit of Intermission field then there is a chance that the CPU reconfigures some Tx MBs and the winner MB is not the best to be transmitted. The optimal configuration for TASD can be calculated as: $TASD = 25 - \{f_{CANCLK} \times [MAXB + 3 - (RFEN \times 8) - (RFEN \times RFFN \times 2)] \times 2\} / \{f_{SYS} \times [1+(PSEG1+1)+(PSEG2+1)+(PROPSEG+1)] \times (PRESDIV+1)\}$ where: • f_{CANCLK} is the Protocol Engine (PE) Clock (see section "Protocol Timing"), in Hz; • f_{SYS} is the peripheral clock, in Hz; • MAXMB is the value in CTRL1[MAXMB] field; • RFEN is the value in CTRL1[RFEN] bit; • RFFN is the value in CTRL2[RFFN] field; • PSEG1 is the value in CTRL1[PSEG1] field; • PSEG2 is the value in CTRL1[PSEG2] field; • PROPSEG is the value in CTRL1[PROPSEG] field; • PRESDIV is the value in CTRL1[PRESDIV] field. See Section "Arbitration process" and Section "Protocol Timing" for more details. **NOTE:** The recommended value for TASD is 22.
18 MRP	Mailboxes Reception Priority If this bit is set the matching process starts from the Mailboxes and if no match occurs the matching continues on the Rx FIFO. This bit can only be written in Freeze mode as it is blocked by hardware in other modes. 0 Matching starts from Rx FIFO and continues on Mailboxes. 1 Matching starts from Mailboxes and continues on Rx FIFO.
17 RRS	Remote Request Storing If this bit is asserted Remote Request Frame is submitted to a matching process and stored in the corresponding Message Buffer in the same fashion of a Data Frame. No automatic Remote Response Frame will be generated. If this bit is negated the Remote Request Frame is submitted to a matching process and an automatic Remote Response Frame is generated if a Message Buffer with CODE=0b1010 is found with the same ID. This bit can only be written in Freeze mode as it is blocked by hardware in other modes. 0 Remote Response Frame is generated. 1 Remote Request Frame is stored.
16 EACEN	Entire Frame Arbitration Field Comparison Enable for Rx Mailboxes This bit controls the comparison of IDE and RTR bits whithin Rx Mailboxes filters with their corresponding bits in the incoming frame by the matching process. This bit does not affect matching for Rx FIFO. This bit can only be written in Freeze mode as it is blocked by hardware in other modes. 0 Rx Mailbox filter's IDE bit is always compared and RTR is never compared despite mask bits. 1 Enables the comparison of both Rx Mailbox filter's IDE and RTR bit with their corresponding bits within the incoming frame. Mask bits do apply.
Reserved	This field is reserved.

Table continues on the next page...

CAN_CTRL2 field descriptions (continued)

Field	Description
	This read-only field is reserved and always has the value 0.

1. The number of the last remaining available mailboxes is defined by the least value between the parameter NUMBER_OF_MB minus 1 and the MCR[MAXMB] field.
2. If Rx Individual Mask Registers are not enabled then all Rx FIFO filters are affected by the Rx FIFO Global Mask.

32.3.15 Error and Status 2 Register (CAN_ESR2)

This register reflects various interrupt flags and some general status.

Address: E800h base + 1Ch offset = E81Ch

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0									LPTM						
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	VPS	IMB	0												
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

CAN_ESR2 field descriptions

Field	Description
31–23 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
22–16 LPTM	Lowest Priority Tx Mailbox If ESR2[VPS] is asserted, this field indicates the lowest number inactive Mailbox (refer to the IMB bit description). If there is no inactive Mailbox then the Mailbox indicated depends on CTRL1[LBUF] bit value. If CTRL1[LBUF] bit is negated then the Mailbox indicated is the one which has the greatest arbitration value (see the "Highest priority Mailbox first" section). If CTRL1[LBUF] bit is asserted then the Mailbox indicated is the highest number active Tx Mailbox. If a Tx Mailbox is being transmitted it is not considered in LPTM calculation. If ESR2[IMB] is not asserted and a frame is transmitted successfully, LPTM is updated with its Mailbox number.
15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
14 VPS	Valid Priority Status This bit indicates whether IMB and LPTM contents are currently valid or not. VPS is asserted upon every complete Tx arbitration process unless the CPU writes to Control and Status word of a Mailbox that has already been scanned (i.e. it is behind Tx Arbitration Pointer) during the Tx arbitration process. If there is no inactive Mailbox and only one Tx Mailbox which is being transmitted then VPS is not asserted. VPS is negated upon the start of every Tx arbitration process or upon a write to Control and Status word of any Mailbox.

Table continues on the next page...

CAN_ESR2 field descriptions (continued)

Field	Description
	NOTE: ESR2[VPS] is not affected by any CPU write into Control Status (C/S) of a MB which is blocked by abort mechanism. When MCR[AEN] is asserted, the abort code write in C/S of a MB that is being transmitted (pending abort), or any write attempt into a Tx MB with IFLAG set is blocked. 0 Contents of IMB and LPTM are invalid. 1 Contents of IMB and LPTM are valid.
13 IMB	Inactive Mailbox If ESR2[VPS] is asserted, this bit indicates whether there is any inactive Mailbox (CODE field is either 0b1000 or 0b0000). This bit is asserted in the following cases: • During arbitration, if an LPTM is found and it is inactive. • If IMB is not asserted and a frame is transmitted successfully. This bit is cleared in all start of arbitration (see Section "Arbitration process"). **NOTE:** LPTM mechanism have the following behavior: if an MB is successfully transmitted and ESR2[IMB]=0 (no inactive Mailbox), then ESR2[VPS] and ESR2[IMB] are asserted and the index related to the MB just transmitted is loaded into ESR2[LPTM]. 0 If ESR2[VPS] is asserted, the ESR2[LPTM] is not an inactive Mailbox. 1 If ESR2[VPS] is asserted, there is at least one inactive Mailbox. LPTM content is the number of the first one.
Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

32.3.16 CRC Register (CAN_CRCR)

This register provides information about the CRC of transmitted messages.

Address: E800h base + 22h offset = E822h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0									MBCRC						
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	TXCRC														
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

CAN_CRCR field descriptions

Field	Description
31–23 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

Table continues on the next page...

CAN_CRCR field descriptions (continued)

Field	Description
22–16 MBCRC	CRC Mailbox This field indicates the number of the Mailbox corresponding to the value in TXCRC field.
15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
TXCRC	CRC Transmitted This field indicates the CRC value of the last message transmitted. This field is updated at the same time the Tx Interrupt Flag is asserted.

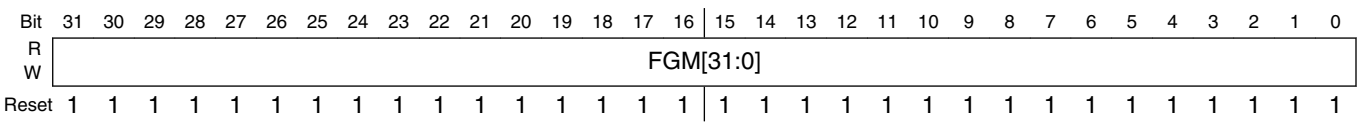
32.3.17 Rx FIFO Global Mask Register (CAN_RXFGMASK)

This register is located in RAM.

If Rx FIFO is enabled RXFGMASK is used to mask the Rx FIFO ID Filter Table elements that do not have a corresponding RXIMR according to CTRL2[RFFN] field setting.

This register can only be written in Freeze mode as it is blocked by hardware in other modes.

Address: E800h base + 24h offset = E824h



CAN_RXFGMASK field descriptions

Field	Description
FGM[31:0]	Rx FIFO Global Mask Bits These bits mask the ID Filter Table elements bits in a perfect alignment. The following table shows how the FGM bits correspond to each IDAF field.

Rx FIFO ID Filter Table Elements Format (MCR[IDAM])	Identifier Acceptance Filter Fields					
	RTR	IDE	RXIDA	RXIDB[1]	RXIDC[2]	Reserved
A	FGM[31]	FGM[30]	FGM[29:1]	-	-	FGM[0]
B	FGM[31], FGM[15]	FGM[30], FGM[14]	-	FGM[29:16], FGM[13:0]		-
C	-	-		-	FGM[31:24], FGM[23:16],	

CAN_RXFGMASK field descriptions (continued)

Field	Description
	Rx FIFO ID Filter Table Elements Format (MCR[IDAM]) / **Identifier Acceptance Filter Fields**

RTR	IDE	RXIDA	RXIDB[1]	RXIDC[2]	Reserved
				FGM[15:8], FGM[7:0]	

1. If MCR[IDAM] field is equivalent to the format B only the fourteen most significant bits of the Identifier of the incoming frame are compared with the Rx FIFO filter.
2. If MCR[IDAM] field is equivalent to the format C only the eight most significant bits of the Identifier of the incoming frame are compared with the Rx FIFO filter.

0 The corresponding bit in the filter is "don't care."
1 The corresponding bit in the filter is checked.

1. If MCR[IDAM] field is equivalent to the format B only the fourteen most significant bits of the Identifier of the incoming frame are compared with the Rx FIFO filter.
2. If MCR[IDAM] field is equivalent to the format C only the eight most significant bits of the Identifier of the incoming frame are compared with the Rx FIFO filter.

32.3.18 Rx FIFO Information Register (CAN_RXFIR)

RXFIR provides information on Rx FIFO.

This register is the port through which the CPU accesses the output of the RXFIR FIFO located in RAM. The RXFIR FIFO is written by the FlexCAN whenever a new message is moved into the Rx FIFO as well as its output is updated whenever the output of the Rx FIFO is updated with the next message. See Section "Rx FIFO" for instructions on reading this register.

Address: E800h base + 26h offset = E826h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0																							IDHIT								
W																																
Reset	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*

* Notes:
- x = Undefined at reset.

CAN_RXFIR field descriptions

Field	Description
31–9 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

Table continues on the next page...

CAN_RXFIR field descriptions (continued)

Field	Description
IDHIT	Identifier Acceptance Filter Hit Indicator This field indicates which Identifier Acceptance Filter was hit by the received message that is in the output of the Rx FIFO. If multiple filters match the incoming message ID then the first matching IDAF found (lowest number) by the matching process is indicated. This field is valid only while the IFLAG[BUF5I] is asserted.

32.3.19 Rx Individual Mask Registers (CAN_RXIMR*n*)

These registers are located in RAM.

RXIMR are used as acceptance masks for ID filtering in Rx MBs and the Rx FIFO. If the Rx FIFO is not enabled, one mask register is provided for each available Mailbox, providing ID masking capability on a per Mailbox basis.

When the Rx FIFO is enabled (MCR[RFEN] bit is asserted), up to 32 Rx Individual Mask Registers can apply to the Rx FIFO ID Filter Table elements on a one-to-one correspondence depending on the setting of CTRL2[RFFN].

RXIMR can only be written by the CPU while the module is in Freeze Mode; otherwise, they are blocked by hardware.

The Individual Rx Mask Registers are not affected by reset and must be explicitly initialized prior to any reception.

Address: E800h base + 440h offset + (2d × i), where i=0d to 15d

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	MI[31:0]																															
Reset	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*

* Notes:
- x = Undefined at reset.

CAN_RXIMR*n* field descriptions

Field	Description
MI[31:0]	Individual Mask Bits Each Individual Mask Bit masks the corresponding bit in both the Mailbox filter and Rx FIFO ID Filter Table element in distinct ways. For Mailbox filters, see the RXMGMASK register description. For Rx FIFO ID Filter Table elements, see the RXFGMASK register description. 0 The corresponding bit in the filter is "don't care." 1 The corresponding bit in the filter is checked.

CAN_RXIMR*n* field descriptions (continued)

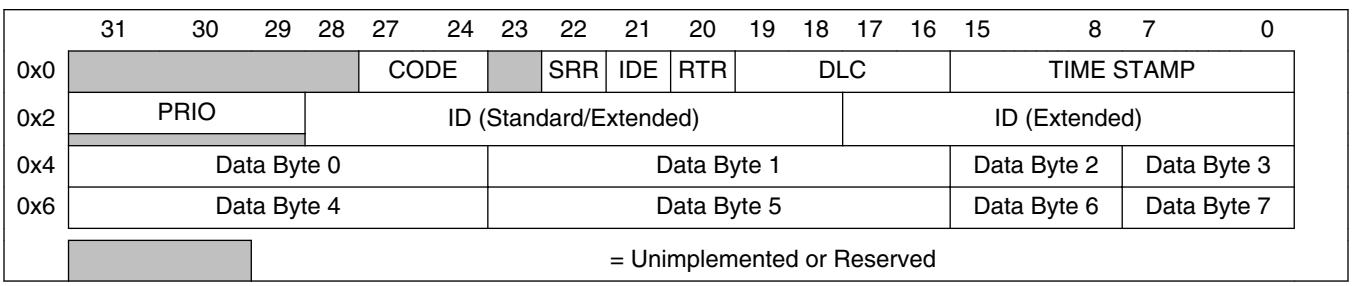
Field	Description

32.3.20 Message buffer structure

The message buffer structure used by the FlexCAN module is represented in the following figure. Both Extended (29-bit identifier) and Standard (11-bit identifier) frames used in the CAN specification (Version 2.0 Part B) are represented. Each individual MB is formed by 16 bytes.

The memory area from 0x40 to 0x0BF is used by the mailboxes.

Table 32-3. Message buffer structure



CODE — Message Buffer Code

This 4-bit field can be accessed (read or write) by the CPU and by the FlexCAN module itself, as part of the message buffer matching and arbitration process. The encoding is shown in Table 32-4 and Table 32-5. See Functional description for additional information.

Table 32-4. Message buffer code for Rx buffers

CODE description	Rx code BEFORE receive new frame	SRV[1]	Rx code AFTER successful reception[2]	RRS[3]	Comment
0b0000: INACTIVE — MB is not active.	INACTIVE	—	—	—	MB does not participate in the matching process.
0b0100: EMPTY — MB is active and empty.	EMPTY	—	FULL	—	When a frame is received successfully (after the Move-in) process), the CODE field is automatically updated to FULL.

Table continues on the next page...

Table 32-4. Message buffer code for Rx buffers (continued)

CODE description	Rx code BEFORE receive new frame	SRV[1]	Rx code AFTER successful reception[2]	RRS[3]	Comment
0b0010: FULL — MB is full.	FULL	Yes	FULL	—	The act of reading the C/S word followed by unlocking the MB (SRV) does not make the code return to EMPTY. It remains FULL. If a new frame is moved to the MB after the MB was serviced, the code still remains FULL. See Matching process for matching details related to FULL code.
		No	OVERRUN	—	If the MB is FULL and a new frame is moved to this MB before the CPU services it, the CODE field is automatically updated to OVERRUN. See Matching process for details about overrun behavior.
0b0110: OVERRUN — MB is being overwritten into a full buffer.	OVERRUN	Yes	FULL	—	If the CODE field indicates OVERRUN and CPU has serviced the MB, when a new frame is moved to the MB then the code returns to FULL.
		No	OVERRUN	—	If the CODE field already indicates OVERRUN, and another new frame must be moved, the MB will be overwritten again, and the code will remain OVERRUN. See Matching process for details about overrun behavior.

Table continues on the next page...

Table 32-4. Message buffer code for Rx buffers (continued)

CODE description	Rx code BEFORE receive new frame	SRV[1]	Rx code AFTER successful reception[2]	RRS[3]	Comment
0b1010: RANSWER[4] — A frame was configured to recognize a Remote Request Frame and transmit a Response Frame in return.	RANSWER	—	TANSWER(0b1110)	0	A Remote Answer was configured to recognize a remote request frame received. After that an MB is set to transmit a response frame. The code is automatically changed to TANSWER (0b1110). See Matching process for details. If CTRL2[RRS] is negated, transmit a response frame whenever a remote request frame with the same ID is received.
		—	—	1	This code is ignored during matching and arbitration process. See Matching process for details.
CODE[0]=1b1: BUSY — FlexCAN is updating the contents of the MB. The CPU must not access the MB.	BUSY[5]	—	FULL	—	Indicates that the MB is being updated. It will be negated automatically and does not interfere with the next CODE.
		—	OVERRUN	—	

1. SRV: Serviced MB. MB was read and unlocked by reading TIMER or other MB.
2. A frame is considered a successful reception after the frame to be moved to MB (move-in process). See Move-in for details.
3. Remote Request Stored bit from CTRL2 register. See section "Control 2 Register (CTRL2)" for details.
4. Code 0b1010 is not considered Tx and an MB with this code should not be aborted.
5. Note that for Tx MBs, the BUSY bit should be ignored upon read, except when AEN bit is set in the MCR register. If this bit is asserted, the corresponding MB does not participate in the matching process.

Table 32-5. Message buffer code for Tx buffers

CODE Description	Tx Code BEFORE tx frame	MB RTR	Tx Code AFTER successful transmission	Comment
0b1000: INACTIVE — MB is not active	INACTIVE	—	—	MB does not participate in arbitration process.

Table continues on the next page...

Table 32-5. Message buffer code for Tx buffers (continued)

CODE Description	Tx Code BEFORE tx frame	MB RTR	Tx Code AFTER successful transmission	Comment
0b1001: ABORT — MB is aborted	ABORT	—	—	MB does not participate in arbitration process.
0b1100: DATA — MB is a Tx Data Frame (MB RTR must be 0)	DATA	0	INACTIVE	Transmit data frame unconditionally once. After transmission, the MB automatically returns to the INACTIVE state.
0b1100: REMOTE — MB is a Tx Remote Request Frame (MB RTR must be 1)	REMOTE	1	EMPTY	Transmit remote request frame unconditionally once. After transmission, the MB automatically becomes an Rx Empty MB with the same ID.
0b1110: TANSWER — MB is a Tx Response Frame from an incoming Remote Request Frame	TANSWER	—	RANSWER	This is an intermediate code that is automatically written to the MB by the CHI as a result of a match to a remote request frame. The remote response frame will be transmitted unconditionally once, and then the code will automatically return to RANSWER (0b1010). The CPU can also write this code with the same effect. The remote response frame can be either a data frame or another remote request frame depending on the RTR bit value. See Matching process and Arbitration process for details.

SRR — Substitute Remote Request

Fixed recessive bit, used only in extended format. It must be set to one by the user for transmission (Tx Buffers) and will be stored with the value received on the CAN bus for Rx receiving buffers. It can be received as either recessive or dominant. If FlexCAN receives this bit as dominant, then it is interpreted as an arbitration loss.

1 = Recessive value is compulsory for transmission in extended format frames

0 = Dominant is not a valid value for transmission in extended format frames

IDE — ID Extended Bit

This field identifies whether the frame format is standard or extended.

1 = Frame format is extended

0 = Frame format is standard

RTR — Remote Transmission Request

This bit affects the behavior of remote frames and is part of the reception filter. See Table 32-4, Table 32-5, and the description of the RRS bit in Control 2 Register (CTRL2) for additional details.

If FlexCAN transmits this bit as '1' (recessive) and receives it as '0' (dominant), it is interpreted as an arbitration loss. If this bit is transmitted as '0' (dominant), then if it is received as '1' (recessive), the FlexCAN module treats it as a bit error. If the value received matches the value transmitted, it is considered a successful bit transmission.

1 = Indicates the current MB may have a remote request frame to be transmitted if MB is Tx. If the MB is Rx then incoming remote request frames may be stored.

0 = Indicates the current MB has a data frame to be transmitted. In Rx MB it may be considered in matching processes.

DLC — Length of Data in Bytes

This 4-bit field is the length (in bytes) of the Rx or Tx data, which is located in offset 0x4 through 0x7 of the MB space (see the Table 32-3). In reception, this field is written by the FlexCAN module, copied from the DLC (Data Length Code) field of the received frame. In transmission, this field is written by the CPU and corresponds to the DLC field value of the frame to be transmitted. When RTR = 1, the frame to be transmitted is a remote frame and does not include the data field, regardless of the DLC field (see Table 32-6).

TIME STAMP — Free-Running Counter Time Stamp

This 16-bit field is a copy of the Free-Running Timer, captured for Tx and Rx frames at the time when the beginning of the Identifier field appears on the CAN bus.

PRIO — Local priority

This 3-bit field is used only when LPRIO_EN bit is set in MCR, and it only makes sense for Tx mailboxes. These bits are not transmitted. They are appended to the regular ID to define the transmission priority. See Arbitration process.

ID — Frame Identifier

In standard frame format, only the 11 most significant bits (28 to 18) are used for frame identification in both receive and transmit cases. The 18 least significant bits are ignored. In extended frame format, all bits are used for frame identification in both receive and transmit cases.

DATA BYTE 0–7 — Data Field

Up to eight bytes can be used for a data frame.

For Rx frames, the data is stored as it is received from the CAN bus. DATA BYTE (*n*) is valid only if *n* is less than DLC as shown in the table below.

For Tx frames, the CPU prepares the data field to be transmitted within the frame.

Table 32-6. DATA BYTEs validity

DLC	Valid DATA BYTEs
0	none
1	DATA BYTE 0
2	DATA BYTE 0–1
3	DATA BYTE 0–2
4	DATA BYTE 0–3
5	DATA BYTE 0–4
6	DATA BYTE 0–5
7	DATA BYTE 0–6
8	DATA BYTE 0–7

32.3.21 Rx FIFO structure

When the MCR[RFEN] bit is set, the memory area from 0x40 to 0x6E (which is normally occupied by MBs 0–5) is used by the reception FIFO engine.

The region 0x40 to 0x46 contains the output of the FIFO which must be read by the CPU as a message buffer. This output contains the oldest message that has been received but not yet read. The region 0x48 to 0x6E is reserved for internal use of the FIFO engine.

An additional memory area, which starts at 0x70 and may extend up to 0x16E (normally occupied by MBs 6–37) depending on the CTRL2[RFFN] field setting, contains the ID filter table (configurable from 8 to 128 table elements) that specifies filtering criteria for accepting frames into the FIFO.

Out of reset, the ID filter table flexible memory area defaults to 0x70 and extends only to 0x7E, which corresponds to MBs 6 to 7 for RFFN = 0, for backward compatibility with previous versions of FlexCAN.

The following shows the Rx FIFO data structure.

Table 32-7. Rx FIFO structure

	31 28 24 23 22 21 20 19 18 17 16 15 8 7 0
0x40	SRR | IDE | RTR | DLC | TIME STAMP
0x42	ID standard | ID extended
0x44	Data byte 0 | Data byte 1 | Data byte 2 | Data byte 3
0x46	Data byte 4 | Data byte 5 | Data byte 6 | Data byte 7
0x48 to 0x6E	Reserved
0x70	ID filter table element 0
0x72	ID filter table element 1
0x74 to 0x16A	ID filter table elements 2 to 125
0x16C	ID filter table element 126
0x16E	ID filter table element 127

= Unimplemented or reserved

Each ID filter table element occupies an entire 32-bit word and can be compounded by one, two, or four Identifier Acceptance Filters (IDAF) depending on the MCR[IDAM] field setting. The following figures show the IDAF indexation.

The following table shows the three different formats of the ID table elements. Note that all elements of the table must have the same format. See Rx FIFO for more information.

Table 32-8. ID table structure

Format	31	30	29 24	23 16	15	14	13 8	7 1	0
A	RTR	IDE	RXIDA (standard = 29–19, extended = 29–1)						
B	RTR	IDE	RXIDB_0 (standard = 29–19, extended = 29–16)		RTR	IDE	RXIDB_1 (standard = 13–3, extended = 13–0)		
C	RXIDC_0 (std/ext = 31–24)		RXIDC_1 (std/ext = 23–16)		RXIDC_2 (std/ext = 15–8)		RXIDC_3 (std/ext = 7–0)		

= Unimplemented or Reserved

RTR — Remote Frame

This bit specifies if Remote Frames are accepted into the FIFO if they match the target ID.

1 = Remote Frames can be accepted and data frames are rejected

0 = Remote Frames are rejected and data frames can be accepted

IDE — Extended Frame

Specifies whether extended or standard frames are accepted into the FIFO if they match the target ID.

1 = Extended frames can be accepted and standard frames are rejected

0 = Extended frames are rejected and standard frames can be accepted

RXIDA — Rx Frame Identifier (Format A)

Specifies an ID to be used as acceptance criteria for the FIFO. In the standard frame format, only the 11 most significant bits (29 to 19) are used for frame identification. In the extended frame format, all bits are used.

RXIDB_0, RXIDB_1 — Rx Frame Identifier (Format B)

Specifies an ID to be used as acceptance criteria for the FIFO. In the standard frame format, the 11 most significant bits (a full standard ID) (29 to 19 and 13 to 3) are used for frame identification. In the extended frame format, all 14 bits of the field are compared to the 14 most significant bits of the received ID.

RXIDC_0, RXIDC_1, RXIDC_2, RXIDC_3 — Rx Frame Identifier (Format C)

Specifies an ID to be used as acceptance criteria for the FIFO. In both standard and extended frame formats, all 8 bits of the field are compared to the 8 most significant bits of the received ID.

32.4 Functional description

The FlexCAN module is a CAN protocol engine with a very flexible mailbox system for transmitting and receiving CAN frames. The mailbox system is composed by a set of Message Buffers (MB) that store configuration and control data, time stamp, message ID and data (see Message buffer structure). The memory corresponding to the first 38 MBs can be configured to support a FIFO reception scheme with a powerful ID filtering mechanism, capable of checking incoming frames against a table of IDs (up to 128 extended IDs or 256 standard IDs or 512 8-bit ID slices), with individual mask register for up to 32 ID Filter Table elements. Simultaneous reception through FIFO and mailbox is supported. For mailbox reception, a matching algorithm makes it possible to store received frames only into MBs that have the same ID programmed on its ID field. A masking scheme makes it possible to match the ID programmed on the MB with a range

of IDs on received CAN frames. For transmission, an arbitration algorithm decides the prioritization of MBs to be transmitted based on the message ID (optionally augmented by 3 local priority bits) or the MB ordering.

Before proceeding with the functional description, an important concept must be explained. A Message Buffer is said to be "active" at a given time if it can participate in both the Matching and Arbitration processes. An Rx MB with a 0b0000 code is inactive (refer to Table 32-4). Similarly, a Tx MB with a 0b1000 or 0b1001 code is also inactive (refer to Table 32-5).

32.4.1 Transmit process

To transmit a CAN frame, the CPU must prepare a Message Buffer for transmission by executing the following procedure:

1. Check whether the respective interrupt bit is set and clear it.
2. If the MB is active (transmission pending), write the ABORT code (0b1001) to the CODE field of the Control and Status word to request an abortion of the transmission. Wait for the corresponding IFLAG to be asserted by polling the IFLAG register or by the interrupt request if enabled by the respective IMASK. Then read back the CODE field to check if the transmission was aborted or transmitted (see Transmission abort mechanism). If backwards compatibility is desired (MCR[AEN] bit is negated), just write the INACTIVE code (0b1000) to the CODE field to inactivate the MB but then the pending frame may be transmitted without notification (see Mailbox inactivation).
3. Write the ID word.
4. Write the data bytes.
5. Write the DLC, Control, and CODE fields of the Control and Status word to activate the MB.

When the MB is activated, it will participate into the arbitration process and eventually be transmitted according to its priority. At the end of the successful transmission, the value of the Free Running Timer is written into the Time Stamp field, the CODE field in the Control and Status word is updated, the CRC Register is updated, a status flag is set in the Interrupt Flag Register and an interrupt is generated if allowed by the corresponding Interrupt Mask Register bit. The new CODE field after transmission depends on the code that was used to activate the MB (see Table 32-4 and Table 32-5 in Message buffer structure).

When the Abort feature is enabled (MCR[AEN] is asserted), after the Interrupt Flag is asserted for a Mailbox configured as transmit buffer, the Mailbox is blocked, therefore the CPU is not able to update it until the Interrupt Flag is negated by CPU. This means that the CPU must clear the corresponding IFLAG before starting to prepare this MB for a new transmission or reception.

32.4.2 Arbitration process

The arbitration process scans the Mailboxes searching the Tx one that holds the message to be sent in the next opportunity. This Mailbox is called the *arbitration winner*.

The scan starts from the lowest number Mailbox and runs toward the higher ones.

The arbitration process is triggered in the following events:

- From the CRC field of the CAN frame. The start point depends on the CTRL2[TASD] field value.
- During the Error Delimiter field of a CAN frame.
- During the Overload Delimiter field of a CAN frame.
- When the winner is inactivated and the CAN bus has still not reached the first bit of the Intermission field.
- When there is CPU write to the C/S word of a winner MB and the CAN bus has still not reached the first bit of the Intermission field.
- When CHI is in Idle state and the CPU writes to the C/S word of any MB.
- When FlexCAN exits Bus Off state.
- Upon leaving Freeze mode or Low Power mode.

If the arbitration process does not manage to evaluate all Mailboxes before the CAN bus has reached the first bit of the Intermission field the temporary arbitration winner is invalidated and the FlexCAN will not compete for the CAN bus in the next opportunity.

The arbitration process selects the winner among the active Tx Mailboxes at the end of the scan according to both CTRL1[LBUF] and MCR[LPRIOEN] bits settings.

32.4.2.1 Lowest-number Mailbox first

If CTRL1[LBUF] bit is asserted the first (lowest number) active Tx Mailbox found is the arbitration winner. MCR[LPRIOEN] bit has no effect when CTRL1[LBUF] is asserted.

32.4.2.2 Highest-priority Mailbox first

If CTRL1[LBUF] bit is negated, then the arbitration process searches the active Tx Mailbox with the highest priority, which means that this Mailbox's frame would have a higher probability to win the arbitration on CAN bus when multiple external nodes compete for the bus at the same time.

The sequence of bits considered for this arbitration is called the *arbitration value* of the Mailbox. The highest-priority Tx Mailbox is the one that has the lowest arbitration value among all Tx Mailboxes.

If two or more Mailboxes have equivalent arbitration values, the Mailbox with the lowest number is the arbitration winner.

The composition of the arbitration value depends on MCR[LPRIOEN] bit setting.

32.4.2.2.1 Local Priority disabled

If MCR[LPRIOEN] bit is negated the arbitration value is built in the exact sequence of bits as they would be transmitted in a CAN frame (see the following table) in such a way that the Local Priority is disabled.

Table 32-9. Composition of the arbitration value when Local Priority is disabled

Format	Mailbox Arbitration Value (32 bits)				
Standard (IDE = 0)	Standard ID (11 bits)	RTR (1 bit)	IDE (1 bit)	- (18 bits)	- (1 bit)
Extended (IDE = 1)	Extended ID[28:18] (11 bits)	SRR (1 bit)	IDE (1 bit)	Extended ID[17:0] (18 bits)	RTR (1 bit)

32.4.2.2.2 Local Priority enabled

If Local Priority is desired MCR[LPRIOEN] must be asserted. In this case the Mailbox PRIO field is included at the very left of the arbitration value (see the following table).

Table 32-10. Composition of the arbitration value when Local Priority is enabled

Format	Mailbox Arbitration Value (35 bits)					
Standard (IDE = 0)	PRIO (3 bits)	Standard ID (11 bits)	RTR (1 bit)	IDE (1 bit)	- (18 bits)	- (1 bit)
Extended (IDE = 1)	PRIO (3 bits)	Extended ID[28:18] (11 bits)	SRR (1 bit)	IDE (1 bit)	Extended ID[17:0] (18 bits)	RTR (1 bit)

As the PRIO field is the most significant part of the arbitration value Mailboxes with low PRIO values have higher priority than Mailboxes with high PRIO values regardless the rest of their arbitration values.

Note that the PRIO field is not part of the frame on the CAN bus. Its purpose is only to affect the internal arbitration process.

32.4.2.3 Arbitration process (continued)

After the arbitration winner is found, its content is copied to a hidden auxiliary MB called Tx Serial Message Buffer (Tx SMB), which has the same structure as a normal MB but is not user accessible. This operation is called move-out and after it is done, write access to the corresponding MB is blocked (if the AEN bit in MCR is asserted). The write access is released in the following events:

- After the MB is transmitted
- FlexCAN enters in Freeze mode or Bus Off
- FlexCAN loses the bus arbitration or there is an error during the transmission

At the first opportunity window on the CAN bus, the message on the Tx SMB is transmitted according to the CAN protocol rules. FlexCAN transmits up to eight data bytes, even if the Data Length Code (DLC) field value is greater than that.

Arbitration process can be triggered in the following situations:

- During Rx and Tx frames from CAN CRC field to end of frame. TASD value may be changed to optimize the arbitration start point.
- During CAN BusOff state from TX_ERR_CNT=124 to 128. TASD value may be changed to optimize the arbitration start point.
- During C/S write by CPU in BusIdle. First C/S write starts arbitration process and a second C/S write during this same arbitration restarts the process. If other C/S writes are performed, Tx arbitration process is pending. If there is no arbitration winner after arbitration process has finished, then TX arbitration machine begins a new arbitration process.

    - If there is a pending arbitration and BusIdle state starts then an arbitration process is triggered. In this case the first and second C/S write in BusIdle will not restart the arbitration process. It is possible that there is not enough time to finish arbitration in WaitForBusIdle state and the next state is Idle. In this case the scan is not interrupted, and it is completed during BusIdle state. During this arbitration C/S write does not cause arbitration restart.
- Arbitration winner deactivation during a valid arbitration window.
- Upon Leave Freeze mode (first bit of the WaitForBusIdle state). If there is a re-synchronization during WaitForBusIdle arbitration process is restarted.

Arbitration process stops in the following situation:

- All Mailboxes were scanned
- A Tx active Mailbox is found in case of Lowest Buffer feature enabled
- Arbitration winner inactivation or abort during any arbitration process
- There was not enough time to finish Tx arbitration process (for instance, when a deactivation was performed near the end of frame). In this case arbitration process is pending.
- Error or Overload flag in the bus
- Low Power or Freeze mode request in Idle state

Arbitration is considered pending as described below:

- It was not possible to finish arbitration process in time
- C/S write during arbitration if write is performed in a MB whose number is lower than the Tx arbitration pointer
- Any C/S write if there is no Tx Arbitration process in progress
- Rx Match has just updated a Rx Code to Tx Code
- Entering Busoff state

C/S write during arbitration has the following effect:

- If C/S write is performed in the arbitration winner, a new process is restarted immediately.
- If C/S write is performed in a MB whose number is higher than the Tx arbitration pointer, the ongoing arbitration process will scan this MB as normal.

32.4.3 Receive process

To be able to receive CAN frames into a Mailbox, the CPU must prepare it for reception by executing the following steps:

1. If the Mailbox is active (either Tx or Rx) inactivate the Mailbox (see Mailbox inactivation), preferably with a safe inactivation (see Transmission abort mechanism).
2. Write the ID word
3. Write the EMPTY code (0b0100) to the CODE field of the Control and Status word to activate the Mailbox.

After the MB is activated, it will be able to receive frames that match the programmed filter. At the end of a successful reception, the Mailbox is updated by the *move-in* process (see Move-in) as follows:

1. The received Data field (8 bytes at most) is stored.
2. The received Identifier field is stored.
3. The value of the Free Running Timer at the time of the second bit of frame's Identifier field is written into the Mailbox's Time Stamp field.
4. The received SRR, IDE, RTR, and DLC fields are stored.
5. The CODE field in the Control and Status word is updated (see Table 32-4 and Table 32-5 in Section Message buffer structure).
6. A status flag is set in the Interrupt Flag Register and an interrupt is generated if allowed by the corresponding Interrupt Mask Register bit.

The recommended way for CPU servicing (read) the frame received in an Mailbox is using the following procedure:

1. Read the Control and Status word of that Mailbox.
2. Check if the BUSY bit is deasserted, indicating that the Mailbox is locked. Repeat step 1) while it is asserted. See Mailbox lock mechanism.
3. Read the contents of the Mailbox. Once Mailbox is locked now, its contents won't be modified by FlexCAN Move-in processes. See Move-in.
4. Acknowledge the proper flag at IFLAG registers.
5. Read the Free Running Timer. It is optional but recommended to unlock Mailbox as soon as possible and make it available for reception.

The CPU should synchronize to frame reception by the status flag bit for the specific Mailbox in one of the IFLAG Registers and not by the CODE field of that Mailbox. Polling the CODE field does not work because once a frame was received and the CPU

services the Mailbox (by reading the C/S word followed by unlocking the Mailbox), the CODE field will not return to EMPTY. It will remain FULL, as explained in Table 32-4 . If the CPU tries to workaround this behavior by writing to the C/S word to force an EMPTY code after reading the Mailbox without a prior *safe inactivation*, a newly received frame matching the filter of that Mailbox may be lost.

CAUTION

In summary: *never do polling by reading directly the C/S word of the Mailboxes. Instead, read the IFLAG registers.*

Note that the received frame's Identifier field is always stored in the matching Mailbox, thus the contents of the ID field in an Mailbox may change if the match was due to masking. Note also that FlexCAN does receive frames transmitted by itself if there exists a matching Rx Mailbox, provided the MCR[SRXDIS] bit is not asserted. If the MCR[SRXDIS] bit is asserted, FlexCAN will not store frames transmitted by itself in any MB, even if it contains a matching MB, and no interrupt flag or interrupt signal will be generated due to the frame reception.

To be able to receive CAN frames through the Rx FIFO, the CPU must enable and configure the Rx FIFO during Freeze mode (see Rx FIFO). Upon receiving the Frames Available in Rx FIFO interrupt (see the description of the IFLAG[BUF5I] "Frames available in Rx FIFO" bit in the IMASK1 register), the CPU should service the received frame using the following procedure:

1. Read the Control and Status word (optional – needed only if a mask was used for IDE and RTR bits)
2. Read the ID field (optional – needed only if a mask was used)
3. Read the Data field
4. Read the RXFIR register (optional)
5. Clear the Frames Available in Rx FIFO interrupt by writing 1 to IFLAG[BUF5I] bit (mandatory – releases the MB and allows the CPU to read the next Rx FIFO entry)

32.4.4 Matching process

The matching process scans the MB memory looking for Rx MBs programmed with the same ID as the one received from the CAN bus. If the FIFO is enabled, the priority of scanning can be selected between Mailboxes and FIFO filters. In any case, the matching starts from the lowest number Message Buffer toward the higher ones. If no match is

found within the first structure then the other is scanned subsequently. In the event that the FIFO is full, the matching algorithm will always look for a matching MB outside the FIFO region.

As the frame is being received, it is stored in a hidden auxiliary MB called Rx Serial Message Buffer (Rx SMB).

The matching process start point depends on the following conditions:

- If the received frame is a remote frame, the start point is the CRC field of the frame
- If the received frame is a data frame with DLC field equal to zero, the start point is the CRC field of the frame
- If the received frame is a data frame with DLC field different than zero, the start point is the DATA field of the frame

If a matching ID is found in the FIFO table or in one of the Mailboxes, the contents of the SMB will be transferred to the FIFO or to the matched Mailbox by the move-in process. If any CAN protocol error is detected then no match results will be transferred to the FIFO or to the matched Mailbox at the end of reception.

The matching process scans all matching elements of both Rx FIFO (if enabled) and active Rx Mailboxes (CODE is EMPTY, FULL, OVERRUN or RANSWER) in search of a successful comparison with the matching elements of the Rx SMB that is receiving the frame on the CAN bus. The SMB has the same structure of a Mailbox. The reception structures (Rx FIFO or Mailboxes) associated with the matching elements that had a successful comparison are the *matched structures*. The *matching winner* is selected at the end of the scan among those matched structures and depends on conditions described ahead. See the following table.

Table 32-11. Matching architecture

Structure	SMB[RTR]	CTRL2[RRS]	CTRL2[EACEN]	MB[IDE]	MB[RTR]	MB[ID[1]]	MB[CODE]
Mailbox	0	-	0	cmp[2]	no_cmp[3]	cmp_msk[4]	EMPTY or FULL or OVERRUN
Mailbox	0	-	1	cmp_msk	cmp_msk	cmp_msk	EMPTY or FULL or OVERRUN
Mailbox	1	0	-	cmp	no_cmp	cmp	RANSWER
Mailbox	1	1	0	cmp	no_cmp	cmp_msk	EMPTY or FULL or OVERRUN

Table continues on the next page...

Table 32-11. Matching architecture (continued)

Structure	SMB[RTR]	CTRL2[RRS]	CTRL2[EACEN]	MB[IDE]	MB[RTR]	MB[ID[1]]	MB[CODE]
Mailbox	1	1	1	cmp_msk	cmp_msk	cmp_msk	EMPTY or FULL or OVERRUN
FIFO[5]	-	-	-	cmp_msk	cmp_msk	cmp_msk	-

1. For Mailbox structure, If SMB[IDE] is asserted, the ID is 29 bits (ID Standard + ID Extended). If SMB[IDE] is negated, the ID is only 11 bits (ID Standard). For FIFO structure, the ID depends on IDAM.
2. cmp: Compares the SMB contents with the MB contents regardless the masks.
3. no_cmp: The SMB contents are not compared with the MB contents
4. cmp_msk: Compares the SMB contents with MB contents taking into account the masks.
5. SMB[IDE] and SMB[RTR] are not taken into account when IDAM is type C.

A reception structure is *free-to-receive* when any of the following conditions is satisfied:

- The CODE field of the Mailbox is EMPTY
- The CODE field of the Mailbox is either FULL or OVERRUN and it has already been serviced (the C/S word was read by the CPU and unlocked as described in Mailbox lock mechanism)
- The CODE field of the Mailbox is either FULL or OVERRUN and an inactivation (see Mailbox inactivation) is performed
- The Rx FIFO is not full

The scan order for Mailboxes and Rx FIFO is from the matching element with lowest number to the higher ones.

The matching winner search for Mailboxes is affected by the MCR[IRMQ] bit. If it is negated the matching winner is the first matched Mailbox regardless if it is free-to-receive or not. If it is asserted, the matching winner is selected according to the priority below:

1. the first free-to-receive matched Mailbox;
2. the last non free-to-receive matched Mailbox.

It is possible to select the priority of scan between Mailboxes and Rx FIFO by the CTRL2[MRP] bit.

If the selected priority is Rx FIFO first:

- If the Rx FIFO is a matched structure and is free-to-receive then the Rx FIFO is the matching winner regardless of the scan for Mailboxes
- Otherwise (the Rx FIFO is not a matched structure or is not free-to-receive), then the matching winner is searched among Mailboxes as described above

If the selected priority is Mailboxes first:

- If a free-to-receive matched Mailbox is found, it is the matching winner regardless the scan for Rx FIFO
- If no matched Mailbox is found, then the matching winner is searched in the scan for the Rx FIFO
- If both conditions above are not satisfied and a non free-to-receive matched Mailbox is found then the matching winner determination is conditioned by the MCR[IRMQ] bit:
  - If MCR[IRMQ] bit is negated the matching winner is the first matched Mailbox
  - If MCR[IRMQ] bit is asserted the matching winner is the Rx FIFO if it is a free-to-receive matched structure, otherwise the matching winner is the last non free-to-receive matched Mailbox

See the following table for a summary of matching possibilities.

Table 32-12. Matching possibilities and resulting reception structures

RFEN	IRMQ	MRP	Matched in MB	Matched in FIFO	Reception structure	Description
No FIFO, only MB, match is always MB first						
0	0	X[1]	None[2]	-[3]	None	Frame lost by no match
0	0	X	Free[4]	-	FirstMB	
0	1	X	None	-	None	Frame lost by no match
0	1	X	Free	-	FirstMb	
0	1	X	NotFree	-	LastMB	Overrun
FIFO enabled, no match in FIFO is as if FIFO does not exist						
1	0	X	None	None[5]	None	Frame lost by no match
1	0	X	Free	None	FirstMB	
1	1	X	None	None	None	Frame lost by no match
1	1	X	Free	None	FirstMb	
1	1	X	NotFree	None	LastMB	Overrun
FIFO enabled, Queue disabled						
1	0	0	X	NotFull[6]	FIFO	
1	0	0	None	Full[7]	None	Frame lost by FIFO full (FIFO Overflow)
1	0	0	Free	Full	FirstMB	
1	0	0	NotFree	Full	FirstMB	
1	0	1	None	NotFull	FIFO	
1	0	1	None	Full	None	Frame lost by FIFO full (FIFO Overflow)

Table continues on the next page...

Table 32-12. Matching possibilities and resulting reception structures (continued)

RFEN	IRMQ	MRP	Matched in MB	Matched in FIFO	Reception structure	Description
1	0	1	Free	X	FirstMB	
1	0	1	NotFree	X	FirtsMb	Overrun
FIFO enabled, Queue enabled						
1	1	0	X	NotFull	FIFO	
1	1	0	None	Full	None	Frame lost by FIFO full (FIFO Overflow)
1	1	0	Free	Full	FirstMB	
1	1	0	NotFree	Full	LastMb	Overrun
1	1	1	None	NotFull	FIFO	
1	1	1	Free	X	FirstMB	
1	1	1	NotFree	NotFull	FIFO	
1	1	1	NotFree	Full	LastMb	Overrun

1. This is a don't care condition.
2. Matched in MB "None" means that the frame has not matched any MB (free-to-receive or non-free-to-receive).
3. This is a forbidden condition.
4. Matched in MB "Free" means that the frame matched at least one MB free-to-receive regardless of whether it has matched MBs non-free-to-receive.
5. Matched in FIFO "None" means that the frame has not matched any filter in FIFO. It is as if the FIFO didn't exist (CTRL2[RFEN]=0).
6. Matched in FIFO "NotFull" means that the frame has matched a FIFO filter and has empty slots to receive it.
7. Matched in FIFO "Full" means that the frame has matched a FIFO filter but couldn't store it because it has no empty slots to receive it.

If a non-safe Mailbox inactivation (see Mailbox inactivation) occurs during matching process and the Mailbox inactivated is the temporary matching winner then the temporary matching winner is invalidated. The matching elements scan is not stopped nor restarted, it continues normally. The consequence is that the current matching process works as if the matching elements compared before the inactivation did not exist, therefore a message may be lost.

Suppose, for example, that the FIFO is disabled, IRMQ is enabled and there are two MBs with the same ID, and FlexCAN starts receiving messages with that ID. Let us say that these MBs are the second and the fifth in the array. When the first message arrives, the matching algorithm will find the first match in MB number 2. The code of this MB is EMPTY, so the message is stored there. When the second message arrives, the matching algorithm will find MB number 2 again, but it is not "free-to-receive", so it will keep looking and find MB number 5 and store the message there. If yet another message with the same ID arrives, the matching algorithm finds out that there are no matching MBs that are "free-to-receive", so it decides to overwrite the last matched MB, which is number 5. In doing so, it sets the CODE field of the MB to indicate OVERRUN.

The ability to match the same ID in more than one MB can be exploited to implement a reception queue (in addition to the full featured FIFO) to allow more time for the CPU to service the MBs. By programming more than one MB with the same ID, received messages will be queued into the MBs. The CPU can examine the Time Stamp field of the MBs to determine the order in which the messages arrived.

Matching to a range of IDs is possible by using ID Acceptance Masks. FlexCAN supports individual masking per MB. Refer to the description of the Rx Individual Mask Registers (RXIMRx). During the matching algorithm, if a mask bit is asserted, then the corresponding ID bit is compared. If the mask bit is negated, the corresponding ID bit is "don't care". Please note that the Individual Mask Registers are implemented in RAM, so they are not initialized out of reset. Also, they can only be programmed while the module is in Freeze mode; otherwise, they are blocked by hardware.

FlexCAN also supports an alternate masking scheme with only four mask registers (RXGMASK, RX14MASK, RX15MASK and RXFGMASK) for backwards compatibility with legacy applications. This alternate masking scheme is enabled when the IRMQ bit in the MCR Register is negated.

32.4.5 Move process

There are two types of move process: move-in and move-out.

32.4.5.1 Move-in

The move-in process is the copy of a message received by an Rx SMB to a Rx Mailbox or FIFO that has matched it. If the move destination is the Rx FIFO, attributes of the message are also copied to the RXFIR FIFO. Each Rx SMB has its own move-in process, but only one is performed at a given time as described ahead. The move-in starts only when the message held by the Rx SMB has a corresponding matching winner (see Matching process) and all of the following conditions are true:

- The CAN bus has reached or let past either:
  - The second bit of Intermission field next to the frame that carried the message that is in the Rx SMB
  - The first bit of an overload frame next to the frame that carried the message that is in the Rx SMB
- There is no ongoing matching process

- The destination Mailbox is not locked by the CPU
- There is no ongoing move-in process from another Rx SMB. If more than one move-in processes are to be started at the same time both are performed and the newest substitutes the oldest.

The term *pending move-in* is used throughout the documentation and stands for a move-to-be that still does not satisfy all of the aforementioned conditions.

The move-in is cancelled and the Rx SMB is able to receive another message if any of the following conditions is satisfied:

- The destination Mailbox is inactivated after the CAN bus has reached the first bit of Intermission field next to the frame that carried the message and its matching process has finished
- There is a previous pending move-in to the same destination Mailbox
- The Rx SMB is receiving a frame transmitted by the FlexCAN itself and the self-reception is disabled (MCR[SRXDIS] bit is asserted)
- Any CAN protocol error is detected

Note that the pending move-in is not cancelled if the module enters Freeze or Low-Power mode. It only stays on hold waiting for exiting Freeze and Low-Power mode and to be unlocked. If an MB is unlocked during Freeze mode, the move-in happens immediately.

The move-in process is the execution by the FlexCAN of the following steps:

1. if the message is destined to the Rx FIFO, push IDHIT into the RXFIR FIFO;
2. reads the words DATA0-3 and DATA4-7 from the Rx SMB;
3. writes it in the words DATA0-3 and DATA4-7 of the Rx Mailbox;
4. reads the words Control/Status and ID from the Rx SMB;
5. writes it in the words Control/Status and ID of the Rx Mailbox, updating the CODE field.

The move-in process is not atomic, in such a way that it is immediately cancelled by the inactivation of the destination Mailbox (see Mailbox inactivation) and in this case the Mailbox may be left partially updated, thus incoherent. The exception is if the move-in destination is an Rx FIFO Message Buffer, then the process cannot be cancelled.

The BUSY Bit (least significant bit of the CODE field) of the destination Message Buffer is asserted while the move-in is being performed to alert the CPU that the Message Buffer content is temporarily incoherent.

32.4.5.2 Move-out

The move-out process is the copy of the content from a Tx Mailbox to the Tx SMB when a message for transmission is available (see Section "Arbitration process"). The move-out occurs in the following conditions:

- The first bit of Intermission field
- During Bus Off state when TX Error Counter is in the 124 to 128 range
- During Bus Idle state
- During Wait For Bus Idle state

The move-out process is not atomic. Only the CPU has priority to access the memory concurrently out of Bus Idle state. In Bus Idle, the move-out has the lowest priority to the concurrent memory accesses.

32.4.6 Data coherence

In order to maintain data coherency and FlexCAN proper operation, the CPU must obey the rules described in Transmit process and Receive process.

32.4.6.1 Transmission abort mechanism

The abort mechanism provides a safe way to request the abortion of a pending transmission. A feedback mechanism is provided to inform the CPU if the transmission was aborted or if the frame could not be aborted and was transmitted instead.

Two primary conditions must be fulfilled in order to abort a transmission:

- MCR[AEN] bit must be asserted
- The first CPU action must be the writing of abort code (0b1001) into the CODE field of the Control and Status word.

The active MBs configured as transmission must be aborted first and then they may be updated. If the abort code is written to a Mailbox that is currently being transmitted, or to a Mailbox that was already loaded into the SMB for transmission, the write operation is blocked and the MB is kept active, but the abort request is captured and kept pending until one of the following conditions are satisfied:

- The module loses the bus arbitration
- There is an error during the transmission
- The module is put into Freeze mode

- The module enters in BusOff state
- There is an overload frame

If none of the conditions above are reached, the MB is transmitted correctly, the interrupt flag is set in the IFLAG register, and an interrupt to the CPU is generated (if enabled). The abort request is automatically cleared when the interrupt flag is set. On the other hand, if one of the above conditions is reached, the frame is not transmitted; therefore, the abort code is written into the CODE field, the interrupt flag is set in the IFLAG, and an interrupt is (optionally) generated to the CPU.

If the CPU writes the abort code before the transmission begins internally, then the write operation is not blocked; therefore, the MB is updated and the interrupt flag is set. In this way the CPU just needs to read the abort code to make sure the active MB was *safely inactivated*. Although the AEN bit is asserted and the CPU wrote the abort code, in this case the MB is inactivated and not aborted, because the transmission did not start yet. One Mailbox is only aborted when the abort request is captured and kept pending until one of the previous conditions are satisfied.

The abort procedure can be summarized as follows:

- CPU checks the corresponding IFLAG and clears it, if asserted.
- CPU writes 0b1001 into the CODE field of the C/S word.
- CPU waits for the corresponding IFLAG indicating that the frame was either transmitted or aborted.
- CPU reads the CODE field to check if the frame was either transmitted (CODE=0b1000) or aborted (CODE=0b1001).
- It is necessary to clear the corresponding IFLAG in order to allow the MB to be reconfigured.

32.4.6.2 Mailbox inactivation

Inactivation is a mechanism provided to protect the Mailbox against updates by the FlexCAN internal processes, thus allowing the CPU to rely on Mailbox data coherence after having updated it, even in Normal mode.

Inactivation of transmission Mailboxes must be performed just when MCR[AEN] bit is deasserted.

If a Mailbox is inactivated, it participates in neither the arbitration process nor the matching process until it is reactivated. See Transmit process and Receive process for more detailed instruction on how to inactivate and reactivate a Mailbox.

To inactivate a Mailbox, the CPU must update its CODE field to INACTIVE (either 0b0000 or 0b1000).

Because the user is not able to synchronize the CODE field update with the FlexCAN internal processes, an inactivation can lead to undesirable results:

- A frame in the bus that matches the filtering of the inactivated Rx Mailbox may be lost without notice, even if there are other Mailboxes with the same filter
- A frame containing the message within the inactivated Tx Mailbox may be transmitted without notice

In order to eliminate such risk and perform a *safe inactivation* the CPU must use the following mechanism along with the inactivation itself:

- For Tx Mailboxes, the Transmission Abort (see Transmission abort mechanism)

The inactivation automatically unlocks the Mailbox (see Mailbox lock mechanism).

NOTE

Message Buffers that are part of the Rx FIFO cannot be inactivated. There is no write protection on the FIFO region by FlexCAN. CPU must maintain data coherency in the FIFO region when RFEN is asserted.

32.4.6.3 Mailbox lock mechanism

Other than Mailbox inactivation, FlexCAN has another data coherence mechanism for the receive process. When the CPU reads the Control and Status word of an Rx MB with codes FULL or OVERRUN, FlexCAN assumes that the CPU wants to read the whole MB in an atomic operation, and therefore it sets an internal lock flag for that MB. The lock is released when the CPU reads the Free Running Timer (global unlock operation), or when it reads the Control and Status word of another MB regardless of its code. A CPU write into C/S word also unlocks the MB, but this procedure is not recommended for normal unlock use because it cancels a pending-move and potentially may lose a received message. The MB locking is done to prevent a new frame to be written into the MB while the CPU is reading it.

NOTE

The locking mechanism applies only to Rx MBs that are not part of FIFO and have a code different than INACTIVE

(0b0000) or EMPTY[1] (0b0100). Also, Tx MBs can not be locked.

Suppose, for example, that the FIFO is disabled and the second and the fifth MBs of the array are programmed with the same ID, and FlexCAN has already received and stored messages into these two MBs. Suppose now that the CPU decides to read MB number 5 and at the same time another message with the same ID is arriving. When the CPU reads the Control and Status word of MB number 5, this MB is locked. The new message arrives and the matching algorithm finds out that there are no "free-to-receive" MBs, so it decides to override MB number 5. However, this MB is locked, so the new message can not be written there. It will remain in the SMB waiting for the MB to be unlocked, and only then will be written to the MB.

If the MB is not unlocked in time and yet another new message with the same ID arrives, then the new message overwrites the one on the SMB and there will be no indication of lost messages either in the CODE field of the MB or in the Error and Status Register.

While the message is being moved-in from the SMB to the MB, the BUSY bit on the CODE field is asserted. If the CPU reads the Control and Status word and finds out that the BUSY bit is set, it should defer accessing the MB until the BUSY bit is negated.

Note

If the BUSY bit is asserted or if the MB is empty, then reading the Control and Status word does not lock the MB.

Inactivation takes precedence over locking. If the CPU inactivates a locked Rx MB, then its lock status is negated and the MB is marked as invalid for the current matching round. Any pending message on the SMB will not be transferred anymore to the MB. An MB is unlocked when the CPU reads the Free Running Timer Register (see Section "Free Running Timer Register (TIMER)"), or the C/S word of another MB.

Lock and unlock mechanisms have the same functionality in both Normal and Freeze modes.

An unlock during Normal or Freeze mode results in the move-in of the pending message. However, the move-in is postponed if an unlock occurs during a low power mode (see Modes of operation) and it will take place only when the module resumes to Normal or Freeze modes.

1. In previous FlexCAN versions, reading the C/S word locked the MB even if it was EMPTY. This behavior is maintained when the IRMQ bit is negated.

32.4.7 Rx FIFO

The Rx FIFO is receive-only and is enabled by asserting the MCR[RFEN] bit. The reset value of this bit is zero to maintain software backward compatibility with previous versions of the module that did not have the FIFO feature. The FIFO is 6-message deep. The memory region occupied by the FIFO structure (both Message Buffers and FIFO engine) is described in Rx FIFO structure. The CPU can read the received messages sequentially, in the order they were received, by repeatedly reading a Message Buffer structure at the output of the FIFO.

The IFLAG[BUF5I] (Frames available in Rx FIFO) is asserted when there is at least one frame available to be read from the FIFO. An interrupt is generated if it is enabled by the corresponding mask bit. Upon receiving the interrupt, the CPU can read the message (accessing the output of the FIFO as a Message Buffer) and the RXFIR register and then clear the interrupt. If there are more messages in the FIFO the act of clearing the interrupt updates the output of the FIFO with the next message and update the RXFIR with the attributes of that message, reissuing the interrupt to the CPU. Otherwise, the flag remains negated. The output of the FIFO is only valid whilst the IFLAG[BUF5I] is asserted.

The IFLAG[BUF6I] (Rx FIFO Warning) is asserted when the number of unread messages within the Rx FIFO is increased to 5 from 4 due to the reception of a new one, meaning that the Rx FIFO is almost full. The flag remains asserted until the CPU clears it.

The IFLAG[BUF7I] (Rx FIFO Overflow) is asserted when an incoming message was lost because the Rx FIFO is full. Note that the flag will not be asserted when the Rx FIFO is full and the message was captured by a Mailbox. The flag remains asserted until the CPU clears it.

Clearing one of those three flags does not affect the state of the other two.

An interrupt is generated if an IFLAG bit is asserted and the corresponding mask bit is asserted too.

A powerful filtering scheme is provided to accept only frames intended for the target application, reducing the interrupt servicing work load. The filtering criteria is specified by programming a table of up to 128 32-bit registers, according to CTRL2[RFFN] setting, that can be configured to one of the following formats (see also Rx FIFO structure):

- Format A: 128 IDAFs (extended or standard IDs including IDE and RTR)
- Format B: 256 IDAFs (standard IDs or extended 14-bit ID slices including IDE and RTR)
- Format C: 512 IDAFs (standard or extended 8-bit ID slices)

Note

A chosen format is applied to all entries of the filter table. It is not possible to mix formats within the table.

Every frame available in the FIFO has a corresponding IDHIT (Identifier Acceptance Filter Hit Indicator) that can be read by accessing the RXFIR register. The RXFIR[IDHIT] field refers to the message at the output of the FIFO and is valid while the IFLAG[BUF5I] flag is asserted. The RXFIR register must be read only before clearing the flag, which guarantees that the information refers to the correct frame within the FIFO.

Up to 32 elements of the filter table are individually affected by the Individual Mask Registers (RXIMRx), according to the setting of CTRL2[RFFN], allowing very powerful filtering criteria to be defined. If the IRMQ bit is negated, then the FIFO filter table is affected by RXFGMASK.

32.4.8 CAN protocol related features

This section describes the CAN protocol related features.

32.4.8.1 Remote frames

Remote frame is a special kind of frame. The user can program a mailbox to be a Remote Request Frame by writing the mailbox as Transmit with the RTR bit set to '1'. After the remote request frame is transmitted successfully, the mailbox becomes a Receive Message Buffer, with the same ID as before.

When a remote request frame is received by FlexCAN, it can be treated in three ways, depending on Remote Request Storing (CTRL2[RRS]) and Rx FIFO Enable (MCR[RFEN]) bits:

- If RRS is negated the frame's ID is compared to the IDs of the Transmit Message Buffers with the CODE field 0b1010. If there is a matching ID, then this mailbox frame will be transmitted. Note that if the matching mailbox has the RTR bit set, then FlexCAN will transmit a remote frame as a response. The received remote request frame is not stored in a receive buffer. It is only used to trigger a transmission of a frame in response. The mask registers are not used in remote frame matching, and all ID bits (except RTR) of the incoming received frame should match. In the case that a remote request frame was received and matched a mailbox, this message buffer

immediately enters the internal arbitration process, but is considered as normal Tx mailbox, with no higher priority. The data length of this frame is independent of the DLC field in the remote frame that initiated its transmission.

- If RRS is asserted the frame's ID is compared to the IDs of the receive mailboxes with the CODE field 0b0100, 0b0010 or 0b0110. If there is a matching ID, then this mailbox will store the remote frame in the same fashion of a data frame. No automatic remote response frame will be generated. The mask registers are used in the matching process.
- If RFEN is asserted FlexCAN will not generate an automatic response for remote request frames that match the FIFO filtering criteria. If the remote frame matches one of the target IDs, it will be stored in the FIFO and presented to the CPU. Note that for filtering formats A and B, it is possible to select whether remote frames are accepted or not. For format C, remote frames are always accepted (if they match the ID). Remote Request Frames are considered as normal frames, and generate a FIFO overflow when a successful reception occurs and the FIFO is already full.

32.4.8.2 Overload frames

FlexCAN does transmit overload frames due to detection of following conditions on CAN bus:

- Detection of a dominant bit in the first/second bit of Intermission
- Detection of a dominant bit at the 7th bit (last) of End of Frame field (Rx frames)
- Detection of a dominant bit at the 8th bit (last) of Error Frame Delimiter or Overload Frame Delimiter

32.4.8.3 Time stamp

The value of the Free Running Timer is sampled at the beginning of the Identifier field on the CAN bus, and is stored at the end of "move-in" in the TIME STAMP field, providing network behavior with respect to time.

The Free Running Timer can be reset upon a specific frame reception, enabling network time synchronization. See the TSYN description in the description of the Control 1 Register (CTRL1).

32.4.8.4 Protocol timing

The following figure shows the structure of the clock generation circuitry that feeds the CAN Protocol Engine (PE) submodule. The clock source bit CLKSRC in the CTRL1 Register defines whether the internal clock is connected to the output of a crystal oscillator (Oscillator Clock) or to the Peripheral Clock. In order to guarantee reliable operation, the clock source should be selected while the module is in Disable Mode (bit MDIS set in the Module Configuration Register).

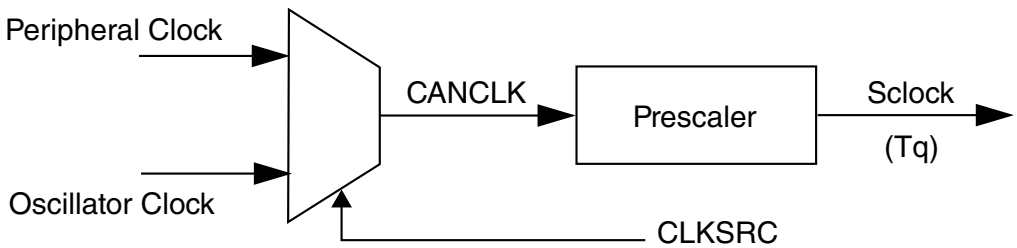


Figure 32-2. CAN engine clocking scheme

The oscillator clock should be selected whenever a tight tolerance (up to 0.1%) is required in the CAN bus timing. The oscillator clock has better jitter performance.

The FlexCAN module supports a variety of means to setup bit timing parameters that are required by the CAN protocol. The Control Register has various fields used to control bit timing parameters: PRESDIV, PROPSEG, PSEG1, PSEG2 and RJW. See the description of the Control 1 Register (CTRL1).

The PRESDIV field controls a prescaler that generates the Serial Clock (Sclock), whose period defines the 'time quantum' used to compose the CAN waveform. A time quantum is the atomic unit of time handled by the CAN engine.

$$f_{Tq} = \frac{f_{CANCLK}}{(\text{Prescaler Value})}$$

A bit time is subdivided into three segments[1] (see Figure 32-3 and Table 32-13):

- SYNC_SEG: This segment has a fixed length of one time quantum. Signal edges are expected to happen within this section

1. For further explanation of the underlying concepts, see ISO/DIS 11519–1, Section 10.3. See also the CAN 2.0A/B protocol specification for bit timing.

- Time Segment 1: This segment includes the Propagation Segment and the Phase Segment 1 of the CAN standard. It can be programmed by setting the PROPSEG and the PSEG1 fields of the CTRL1 Register so that their sum (plus 2) is in the range of 4 to 16 time quanta
- Time Segment 2: This segment represents the Phase Segment 2 of the CAN standard. It can be programmed by setting the PSEG2 field of the CTRL1 Register (plus 1) to be 2 to 8 time quanta long

$$\text{Bit Rate} = \frac{f_{Tq}}{(\text{number of Time Quanta})}$$

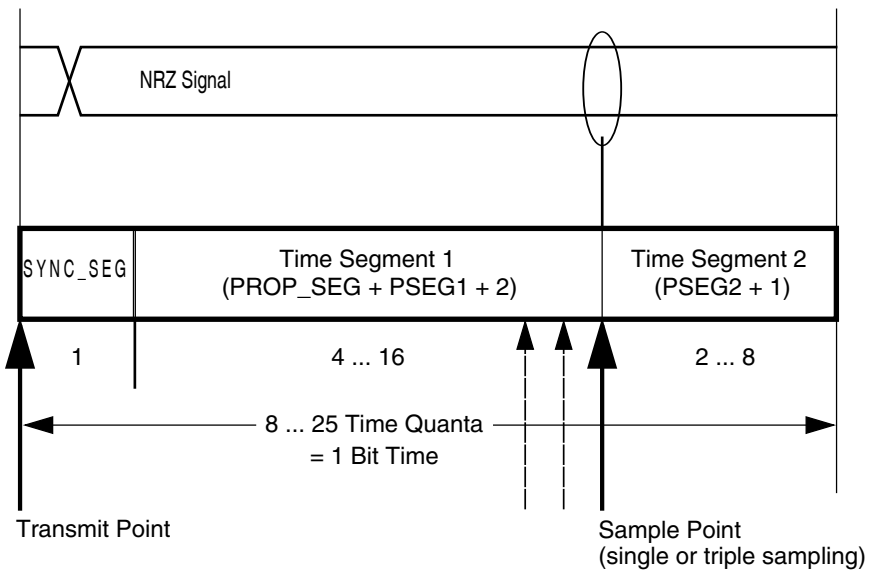


Figure 32-3. Segments within the bit time

Whenever CAN bit is used as a measure of duration (e.g. MCR[FRZACK] and MCR[LPMACK]), the number of peripheral clocks in one CAN bit can be calculated as:

$$NCCP = \frac{f_{SYS} \times [1 + (PSEG1 + 1) + (PSEG2 + 1) + (PROPSEG + 1)] \times (PRESDIV + 1)}{f_{CANCLK}}$$

where:

- NCCP is the number of peripheral clocks in one CAN bit;
- f_{CANCLK} is the Protocol Engine (PE) Clock (see Figure "CAN Engine Clocking Scheme"), in Hz;
- f_{SYS} is the frequency of operation of the system (CHI) clock, in Hz;
- PSEG1 is the value in CTRL1[PSEG1] field;
- PSEG2 is the value in CTRL1[PSEG2] field;

- PROPSEG is the value in CTRL1[PROPSEG] field;
- PRESDIV is the value in CTRL1[PRESDIV] field.

For example, 180 CAN bits = 180 x NCCP peripheral clock periods.

Table 32-13. Time segment syntax

Syntax	Description
SYNC_SEG	System expects transitions to occur on the 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 samples the bus at this point. If the three samples per bit option is selected, then this point marks the position of the third sample.

The following table gives an overview of the CAN compliant segment settings and the related parameter values.

Table 32-14. Bosch CAN 2.0B standard compliant bit time segment settings

Time segment 1	Time segment 2	Re-synchronization jump width
5 .. 10	2	1 .. 2
4 .. 11	3	1 .. 3
5 .. 12	4	1 .. 4
6 .. 13	5	1 .. 4
7 .. 14	6	1 .. 4
8 .. 15	7	1 .. 4
9 .. 16	8	1 .. 4

Note

The user must ensure the bit time settings are in compliance with the CAN Protocol standard (ISO 11898-1). For bit time calculations, use an IPT (Information Processing Time) of 2, which is the value implemented in the FlexCAN module.

32.4.8.5 Arbitration and matching timing

During normal reception and transmission of frames, the matching, arbitration, move-in and move-out processes are executed during certain time windows inside the CAN frame, as shown in the following figures.

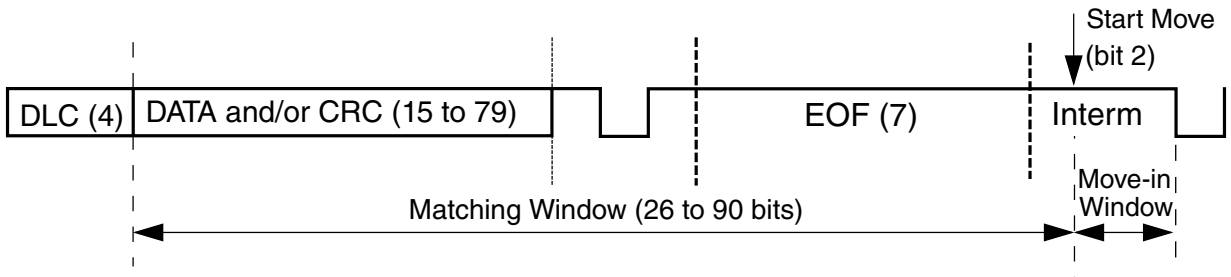


Figure 32-4. Matching and move-in time windows

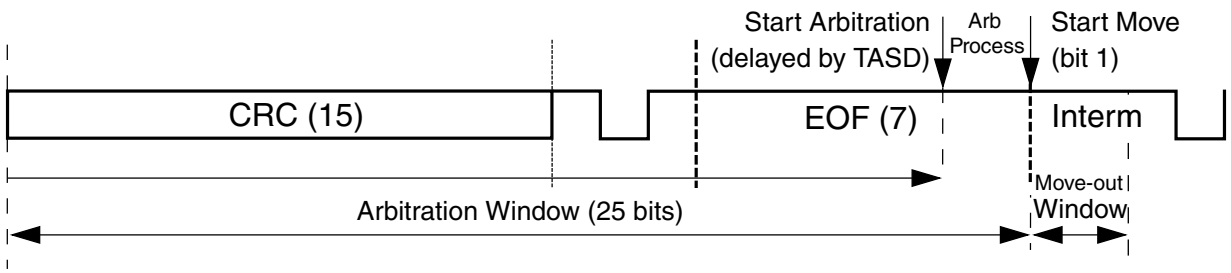


Figure 32-5. Arbitration and move-out time windows

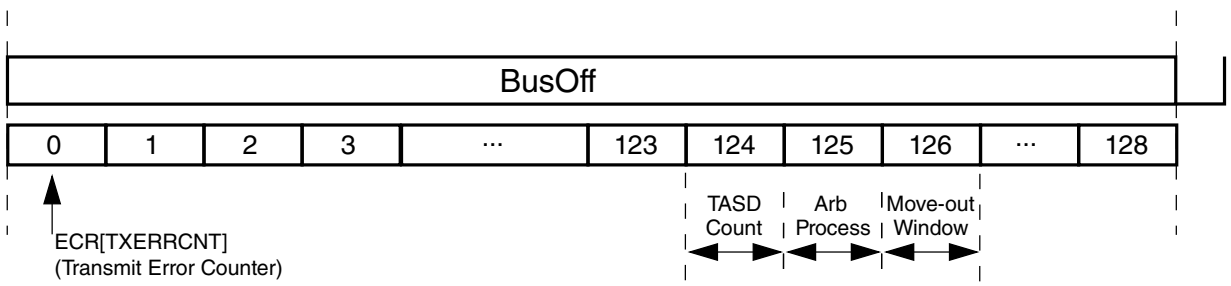


Figure 32-6. Arbitration at the end of bus off and move-out time windows

NOTE

The matching and arbitration timing shown in the preceding figures do not take into account the delay caused by the concurrent memory access due to the CPU or other internal FlexCAN sub-blocks.

When doing matching and arbitration, FlexCAN needs to scan the whole Message Buffer memory during the available time slot. In order to have sufficient time to do that, the following requirements must be observed:

- A valid CAN bit timing must be programmed, as indicated in Table 32-14
- The peripheral clock frequency can not be smaller than the oscillator clock frequency, see Clock domains and restrictions
- There must be a minimum ratio between the peripheral clock frequency and the CAN bit rate, as specified in the following table:

Table 32-15. Minimum ratio between peripheral clock frequency and CAN bit rate

Number of Message Buffers	RFEN	Minimum number of peripheral clocks per CAN bit
16 and 32	0	16
64	0	25
16	1	16
32	1	17
64	1	30

A direct consequence of the first requirement is that the minimum number of time quanta per CAN bit must be 8, therefore the oscillator clock frequency should be at least 8 times the CAN bit rate. The minimum frequency ratio specified in the preceding table can be achieved by choosing a high enough peripheral clock frequency when compared to the oscillator clock frequency, or by adjusting one or more of the bit timing parameters (PRESDIV, PROPSEG, PSEG1, PSEG2) contained in the Control 1 Register (CTRL1).

In case of synchronous operation (when the peripheral clock frequency is equal to the oscillator clock frequency), the number of peripheral clocks per CAN bit can be adjusted by selecting an adequate value for PRESDIV in order to meet the requirement in the preceding table. In case of asynchronous operation (the peripheral clock frequency greater than the oscillator clock frequency), the number of peripheral clocks per CAN bit can be adjusted by both PRESDIV and/or the frequency ratio.

As an example, taking the case of 64 MBs, if the oscillator and peripheral clock frequencies are equal and the CAN bit timing is programmed to have 8 time quanta per bit, then the prescaler factor (PRESDIV + 1) should be at least 2. For prescaler factor equal to one and CAN bit timing with 8 time quanta per bit, the ratio between peripheral and oscillator clock frequencies should be at least 2.

32.4.9 Clock domains and restrictions

The FlexCAN module has two clock domains asynchronous to each other:

- The Bus Domain feeds the Control Host Interface (CHI) submodule and is derived from the peripheral clock.
- The Oscillator Domain feeds the CAN Protocol Engine (PE) submodule and is derived directly from a crystal oscillator clock, so that very low jitter performance can be achieved on the CAN bus.

When CTRL1[CLKSRC] bit is set, synchronous operation occurs because both domains are connected to the peripheral clock (creating a 1:1 ratio between the peripheral and oscillator domain clocks).

When the two domains are connected to clocks with different frequencies and/or phases, there are restrictions on the frequency relationship between the two clock domains. In the case of asynchronous operation, the Bus Domain clock frequency must always be greater than the Oscillator Domain clock frequency.

NOTE

Asynchronous operation with a 1:1 ratio between peripheral and oscillator clocks is not allowed.

32.4.10 Modes of operation details

The FlexCAN module has functional modes and low-power modes. See Modes of operation for an introductory description of all the modes of operation. The following sub-sections contain functional details on Freeze mode and the low-power modes.

CAUTION

"Permanent Dominant" failure on CAN Bus line is not supported by FlexCAN. If a Low-Power request or Freeze mode request is done during a "Permanent Dominant", the corresponding acknowledge can never be asserted.

32.4.10.1 Freeze mode

This mode is requested by the CPU through the assertion of the HALT bit in the MCR Register or when the MCU is put into Debug mode. In both cases it is also necessary that the FRZ bit is asserted in the MCR Register and the module is not in a low-power mode. The acknowledgement is obtained through the assertion by the FlexCAN of FRZ_ACK bit in the same register. The CPU must only consider the FlexCAN in Freeze mode when both request and acknowledgement conditions are satisfied.

When Freeze mode is requested during transmission or reception, FlexCAN does the following:

- Waits to be in either Intermission, Passive Error, Bus Off or Idle state
- Waits for all internal activities like arbitration, matching, move-in and move-out to finish. A pending move-in is not taken into account.
- Ignores the Rx input pin and drives the Tx pin as recessive
- Stops the prescaler, thus halting all CAN protocol activities

- Grants write access to the Error Counters Register, which is read-only in other modes
- Sets the NOT_RDY and FRZ_ACK bits in MCR

After requesting Freeze mode, the user must wait for the FRZ_ACK bit to be asserted in MCR before executing any other action, otherwise FlexCAN may operate in an unpredictable way. In Freeze mode, all memory mapped registers are accessible, except for CTRL1[CLK_SRC] bit that can be read but cannot be written.

Exiting Freeze mode is done in one of the following ways:

- CPU negates the FRZ bit in the MCR Register
- The MCU is removed from Debug Mode and/or the HALT bit is negated

The FRZ_ACK bit is negated after the protocol engine recognizes the negation of the freeze request. When out of Freeze mode, FlexCAN tries to re-synchronize to the CAN bus by waiting for 11 consecutive recessive bits.

32.4.10.2 Module Disable mode

This low power mode is normally used to temporarily disable a complete FlexCAN block, with no power consumption. It is requested by the CPU through the assertion of the MDIS bit in the MCR Register and the acknowledgement is obtained through the assertion by the FlexCAN of the LPM_ACK bit in the same register. The CPU must only consider the FlexCAN in Disable mode when both request and acknowledgement conditions are satisfied.

If the module is disabled during Freeze mode, it requests to disable the clocks to the PE and CHI sub-modules, sets the LPM_ACK bit and negates the FRZ_ACK bit.

If the module is disabled during transmission or reception, FlexCAN does the following:

- Waits to be in either Idle or Bus Off state, or else waits for the third bit of Intermission and then checks it to be recessive
- Waits for all internal activities like arbitration, matching, move-in and move-out to finish. A pending move-in is not taken into account.
- Ignores its Rx input pin and drives its Tx pin as recessive
- Shuts down the clocks to the PE and CHI sub-modules
- Sets the NOTRDY and LPMACK bits in MCR

The Bus Interface Unit continues to operate, enabling the CPU to access memory mapped registers, except the Rx Mailboxes Global Mask Registers, the Rx Buffer 14 Mask Register, the Rx Buffer 15 Mask Register, the Rx FIFO Global Mask Register. The Rx FIFO Information Register, the Message Buffers, the Rx Individual Mask Registers, and the reserved words within RAM may not be accessed when the module is in Disable Mode. Exiting from this mode is done by negating the MDIS bit by the CPU, which causes the FlexCAN to request to resume the clocks and negate the LPM_ACK bit after the CAN protocol engine recognizes the negation of disable mode requested by the CPU.

32.4.10.3 Doze mode

This is a system low power mode in which the CPU bus is kept alive and a global Doze mode request is sent to all peripherals asking them to enter low-power mode. When Doze mode is globally requested, the DOZE bit in MCR Register needs to have been asserted previously for Doze mode to be triggered. The acknowledgement is obtained through the assertion by the FlexCAN of the LPMACK bit in the same register. The CPU must only consider the FlexCAN in Doze mode when both request and acknowledgement conditions are satisfied.

If Doze mode is triggered during Freeze mode, FlexCAN requests to shut down the clocks to the PE and CHI sub-modules, sets the LPMACK bit and negates the FRZACK bit. If Doze Mode is triggered during transmission or reception, FlexCAN does the following:

- Waits to be in either Idle or Bus Off state, or else waits for the third bit of Intermission and checks it to be recessive
- Waits for all internal activities like arbitration, matching, move-in and move-out to finish. A pending move-in is not taken into account.
- Ignores its Rx input pin and drives its Tx pin as recessive
- Shuts down the clocks to the PE and CHI sub-modules
- Sets the NOTRDY and LPMACK bits in MCR

The Bus Interface Unit continues to operate, enabling the CPU to access memory mapped registers, except the Rx Mailboxes Global Mask Registers, the Rx Buffer 14 Mask Register, the Rx Buffer 15 Mask Register, the Rx FIFO Global Mask Register. The Rx FIFO Information Register, the Message Buffers, the Rx Individual Mask Registers, and the reserved words within RAM may not be accessed when the module is in Doze Mode.

Exiting Doze mode is done in one of the following ways:

- CPU removing the Doze mode request
- CPU negating the DOZE bit of the MCR Register
- Self Wake mechanism

In the Self Wake mechanism, if the SLFWAK bit in MCR Register was set at the time FlexCAN entered Doze mode, then upon detection of a recessive to dominant transition on the CAN bus, FlexCAN negates the DOZE bit, requests to resume its clocks and negates the LPMACK after the CAN protocol engine recognizes the negation of the Doze mode request. It also sets the WAKINT bit in the ESR Register and, if enabled by the WAKMSK bit in MCR, generates a Wake Up interrupt to the CPU. FlexCAN will then wait for 11 consecutive recessive bits to synchronize to the CAN bus. As a consequence, it will not receive the frame that woke it up. The following table details the effect of SLFWAK and WAKMSK upon wake-up from Doze mode.

Table 32-16. Wake-up from Doze mode

SLFWAK	WAKINT	WAKMSK	FlexCAN clocks enabled	Wake-up interrupt generated
0	-	-	No	No
0	-	-	No	No
1	0	0	No	No
1	0	1	No	No
1	1	0	Yes	No
1	1	1	Yes	Yes

The sensitivity to CAN bus activity can be modified by applying a low-pass filter function to the Rx CAN input line while in Doze Mode. See the WAKSRC bit in the description of the Module Configuration Register (MCR). This feature can be used to protect FlexCAN from waking up due to short glitches on the CAN bus lines. Such glitches can result from electromagnetic interference within noisy environments.

32.4.10.4 Stop mode

This is a system low-power mode in which all MCU clocks can be stopped for maximum power savings. The Stop mode is globally requested by the CPU and the acknowledgement is obtained through the assertion by the FlexCAN of a Stop Acknowledgement signal. The CPU must only consider the FlexCAN in Stop mode when both request and acknowledgement conditions are satisfied.

If FlexCAN receives the global Stop mode request during Freeze mode, it sets the LPMACK bit, negates the FRZACK bit and then sends the Stop Acknowledge signal to the CPU, in order to shut down the clocks globally.

If Stop mode is requested during transmission or reception, FlexCAN does the following:

- Waits to be in either Idle or Bus Off state, or else waits for the third bit of Intermission and checks it to be recessive
- Waits for all internal activities like arbitration, matching, move-in and move-out to finish. A pending move-in is not taken into account.
- Ignores its Rx input pin and drives its Tx pin as recessive
- Sets the NOTRDY and LPMACK bits in MCR
- Sends a Stop Acknowledge signal to the CPU, so that it can shut down the clocks globally

Stop mode is exited when the CPU resumes the clocks and removes the Stop Mode request. This can be as a result of the Self Wake mechanism.

In the Self Wake mechanism, if the SLFWAK bit in MCR Register was set at the time FlexCAN entered Stop mode, then upon detection of a recessive to dominant transition on the CAN bus, FlexCAN sets the WAKINT bit in the ESR Register and, if enabled by the WAKMSK bit in MCR, generates a Wake Up interrupt to the CPU. Upon receiving the interrupt, the CPU should resume the clocks and remove the Stop mode request. FlexCAN will then wait for 11 consecutive recessive bits to synchronize to the CAN bus. As a consequence, it will not receive the frame that woke it up. The following table details the effect of SLFWAK and WAKMSK upon wake-up from Stop mode. Note that wake-up from Stop mode only works when both bits are asserted.

After the CAN protocol engine recognizes the negation of the Stop mode request, the FlexCAN negates the LPMACK bit. FlexCAN will then wait for 11 consecutive recessive bits to synchronize to the CAN bus. As a consequence, it will not receive the frame that woke it up.

Table 32-17. Wake-up from Stop Mode

SLFWAK	WAKINT	WAKMSK	MCU clocks enabled	Wake-up interrupt generated
0	-	-	No	No
0	-	-	No	No
1	0	0	No	No
1	0	1	No	No

Table continues on the next page...

Table 32-17. Wake-up from Stop Mode (continued)

SLFWAK	WAKINT	WAKMSK	MCU clocks enabled	Wake-up interrupt generated
1	1	0	No	No
1	1	1	Yes	Yes

The sensitivity to CAN bus activity can be modified by applying a low-pass filter function to the Rx CAN input line while in Stop mode. See the WAKSRC bit in in the description of the Module Configuration Register (MCR). This feature can be used to protect FlexCAN from waking up due to short glitches on the CAN bus lines. Such glitches can result from electromagnetic interference within noisy environments.

32.4.11 Interrupts

The module has many interrupt sources: interrupts due to message buffers and interrupts due to the ORed interrupts from MBs, Bus Off, Error, Wake Up, Tx Warning, and Rx Warning.

Each one of the message buffers can be an interrupt source, if its corresponding IMASK bit is set. There is no distinction between Tx and Rx interrupts for a particular buffer, under the assumption that the buffer is initialized for either transmission or reception. Each of the buffers has assigned a flag bit in the IFLAG Registers. The bit is set when the corresponding buffer completes a successful transfer and is cleared when the CPU writes it to 1 (unless another interrupt is generated at the same time).

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.

If the Rx FIFO is enabled (MCR[RFEN] = 1), the interrupts corresponding to MBs 0 to 7 have different meanings. Bit 7 of the IFLAG1 becomes the "FIFO Overflow" flag; bit 6 becomes the FIFO Warning flag, bit 5 becomes the "Frames Available in FIFO flag" and bits 4-0 are unused. See the description of the Interrupt Flags 1 Register (IFLAG1) for more information.

For a combined interrupt where multiple MB interrupt sources are OR'd together, the interrupt is generated when any of the associated MBs (or FIFO, if applicable) generates an interrupt. In this case, the CPU must read the IFLAG registers to determine which MB or FIFO source caused the interrupt.

The interrupt sources for Bus Off, Error, Wake Up, Tx Warning and Rx Warning generate interrupts like the MB interrupt sources, and can be read from both the Error and Status Register 1 and 2. The Bus Off, Error, Tx Warning, and Rx Warning interrupt mask bits are located in the Control 1 Register; the Wake-Up interrupt mask bit is located in the MCR.

32.4.12 Bus interface

The CPU access to FlexCAN registers are subject to the following rules:

- Unrestricted read and write access to supervisor registers (registers identified with S/U in Table "Module Memory Map" in Supervisor Mode or with S only) results in access error.
- Read and write access to implemented reserved address space results in access error.
- Write access to positions whose bits are all currently read-only results in access error. If at least one of the bits is not read-only then no access error is issued. Write permission to positions or some of their bits can change depending on the mode of operation or transitory state. Refer to register and bit descriptions for details.
- Read and write access to unimplemented address space results in access error.
- Read and write access to RAM located positions during Low Power Mode results in access error.
- If MAXMB is programmed with a value smaller than the available number of MBs, then the unused memory space can be used as general purpose RAM space. Note that reserved words within RAM cannot be used. As an example, suppose FlexCAN is configured with 16 MBs, RFFN is 0x0, and MAXMB is programmed with zero. The maximum number of MBs in this case becomes one. The RAM starts at 0x0080, and the space from 0x0080 to 0x008F is used by the one MB. The memory space from 0x0090 to 0x017F is available. The space between 0x0180 and 0x087F is reserved. The space from 0x0880 to 0x0883 is used by the one Individual Mask and the available memory in the Mask Registers space would be from 0x0884 to 0x08BF. From 0x08C0 through 0x09DF there are reserved words for internal use which cannot be used as general purpose RAM. As a general rule, free memory space for general purpose depends only on MAXMB.

32.5 Initialization/application information

This section provide instructions for initializing the FlexCAN module.

32.5.1 FlexCAN initialization sequence

The FlexCAN module may be reset as follows:

- MCU level hard reset, which resets all memory mapped registers asynchronously
- SOFTRST bit in MCR, which resets some of the memory mapped registers synchronously. See Table 32-2 to see what registers are affected by soft reset.
- MCU level soft reset, which has the same effect as the SOFTRST bit in MCR

Soft reset is synchronous and has to follow an internal request/acknowledge procedure across clock domains. Therefore, it may take some time to fully propagate its effects. The SOFTRST bit remains asserted while soft reset is pending, so software can poll this bit to know when the reset has completed. Also, soft reset can not be applied while clocks are shut down in a low power mode. The low power mode should be exited and the clocks resumed before applying soft reset.

The clock source (CLKSRC bit) should be selected while the module is in Disable mode. After the clock source is selected and the module is enabled (MDIS bit negated), FlexCAN automatically goes to Freeze mode. In Freeze mode, FlexCAN is un-synchronized to the CAN bus, the HALT and FRZ bits in MCR Register are set, the internal state machines are disabled and the FRZACK and NOTRDY bits in the MCR Register are set. The Tx pin is in recessive state and FlexCAN does not initiate any transmission or reception of CAN frames. Note that the Message Buffers and the Rx Individual Mask Registers are not affected by reset, so they are not automatically initialized.

For any configuration change/initialization it is required that FlexCAN is put into Freeze mode; see Freeze mode. The following is a generic initialization sequence applicable to the FlexCAN module:

- Initialize the Module Configuration Register
  - Enable the individual filtering per MB and reception queue features by setting the IRMQ bit
  - Enable the warning interrupts by setting the WRNEN bit

    - If required, disable frame self reception by setting the SRXDIS bit
    - Enable the Rx FIFO by setting the RFEN bit
    - Enable the abort mechanism by setting the AEN bit
    - Enable the local priority feature by setting the LPRIOEN bit
- Initialize the Control Register
    - Determine the bit timing parameters: PROPSEG, PSEG1, PSEG2, RJW
    - Determine the bit rate by programming the PRESDIV field
    - Determine the internal arbitration mode (LBUF bit)
- Initialize the Message Buffers
    - The Control and Status word of all Message Buffers must be initialized
    - If Rx FIFO was enabled, the ID filter table must be initialized
    - Other entries in each Message Buffer should be initialized as required
- Initialize the Rx Individual Mask Registers
- Set required interrupt mask bits in the IMASK Registers (for all MB interrupts), in MCR Register for Wake-Up interrupt and in CTRL Register (for Bus Off and Error interrupts)
- Negate the HALT bit in MCR

Starting with the last event, FlexCAN attempts to synchronize to the CAN bus.

Chapter 33
Queued Serial Communications Interface (QSCI)

33.1 Introduction

The SCI allows asynchronous serial communications with peripheral devices.

33.1.1 Features

- Full-duplex or single-wire operation
- Standard mark/space non-return-to-zero (NRZ) format
- 13-bit integer and 3-bit fractional baud rate selection
- Programmable 8-bit or 9-bit data format
- Separately enabled transmitter and receiver
- Separate receiver and transmitter core interrupt requests
- Programmable polarity for transmitter and receiver
- Two receiver wake-up methods: idle line or address mark
- Clockless receiver wake-up on active input edge
- Interrupt-driven operation with multiple flags:
    - Transmitter empty
    - Transmitter idle
    - Receiver full
    - Receiver overrun
    - Receiver input edge

- Noise error
- Framing error
- Parity error

- Receiver framing error detection
- Hardware parity checking
- 1/16 bit-time noise detection

33.1.2 SCI Block Diagram

The following figure shows the SCI block diagram.

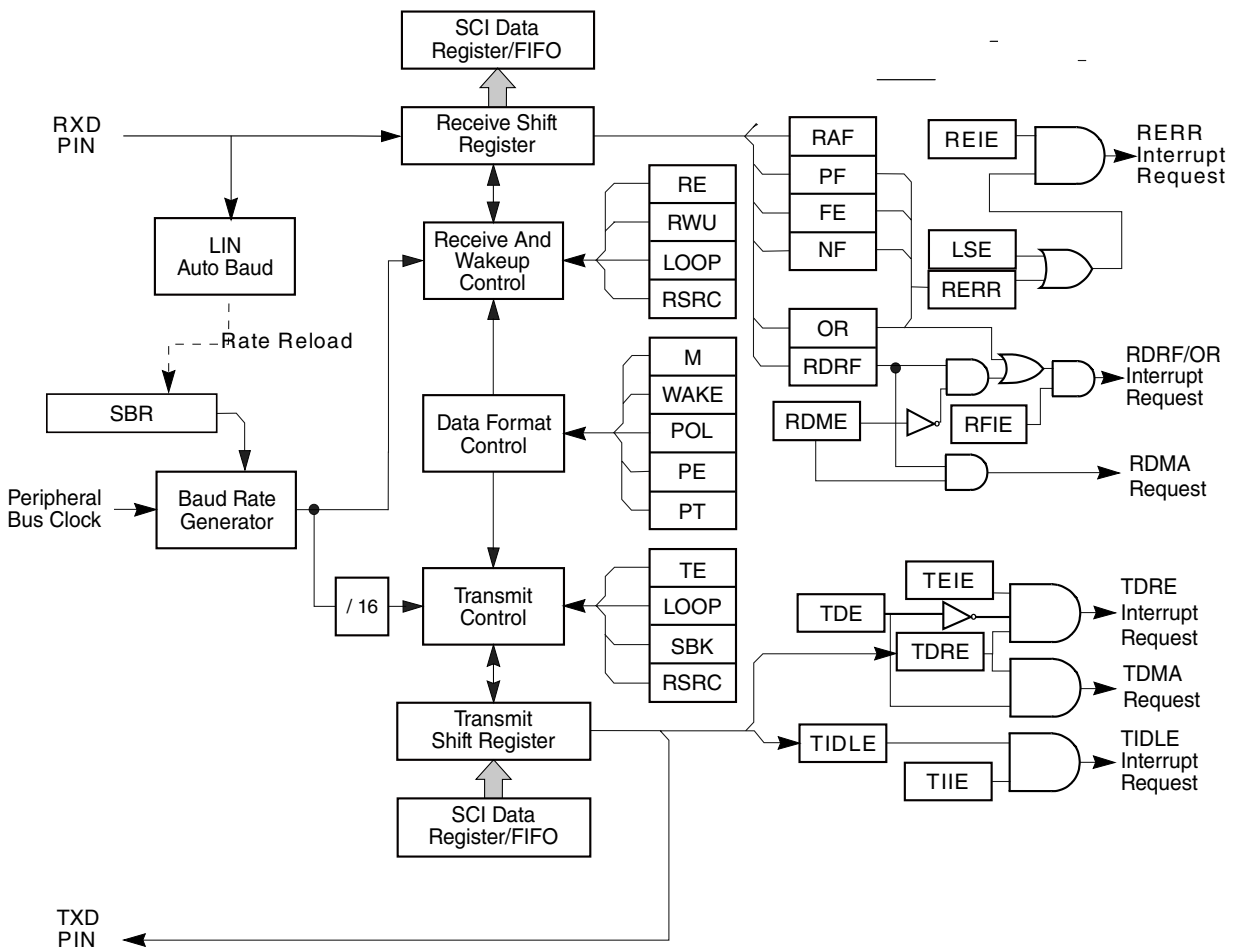


Figure 33-1. SCI Block Diagram with DMA

33.2 External Signal Descriptions

Table 33-1. Signal Properties

Name	I/O Type	Function	Reset State
TXD	O	Transmit Data Pin	1
RXD	I	Receive Data Pin	—

33.2.1 TXD —Transmit Data

The Transmit Data Pin (TXD) is the SCI transmitter pin. TXD is available for general purpose I/O when it is not configured for transmitter operation, such as when CTRL1[TE] = 0.

33.2.2 RXD —Receiver Data

The Receiver Data Pin (RXD) is the SCI receiver pin. RXD is available for general-purpose I/O when it is not configured for receiver operation, such as when CTRL1[RE] = 0.

33.3 Memory Map and Registers

QSCI memory map

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
E080	QSCI Baud Rate Register (QSCI0_RATE)	16	R/W	0200h	33.3.1/866
E081	QSCI Control Register 1 (QSCI0_CTRL1)	16	R/W	0000h	33.3.2/866
E082	QSCI Control Register 2 (QSCI0_CTRL2)	16	R/W	0000h	33.3.3/869
E083	QSCI Status Register (QSCI0_STAT)	16	R	C000h	33.3.4/871
E084	QSCI Data Register (QSCI0_DATA)	16	R/W	0000h	33.3.5/874
E085	QSCI Control Register 3 (QSCI0_CTRL3)	16	R/W	0000h	33.3.6/875
E090	QSCI Baud Rate Register (QSCI1_RATE)	16	R/W	0200h	33.3.1/866
E091	QSCI Control Register 1 (QSCI1_CTRL1)	16	R/W	0000h	33.3.2/866
E092	QSCI Control Register 2 (QSCI1_CTRL2)	16	R/W	0000h	33.3.3/869
E093	QSCI Status Register (QSCI1_STAT)	16	R	C000h	33.3.4/871

Table continues on the next page...

QSCI memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
E094	QSCI Data Register (QSCI1_DATA)	16	R/W	0000h	33.3.5/874
E095	QSCI Control Register 3 (QSCI1_CTRL3)	16	R/W	0000h	33.3.6/875

33.3.1 QSCI Baud Rate Register (QSCIx_RATE)

Read: anytime

Write: anytime

Address: Base address + 0h offset

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read Write	SBR													FRAC_SBR		
Reset	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0

QSCIx_RATE field descriptions

Field	Description
15–3 SBR	SCI Baud Rate divider, a value from 1 to 8191 Refer to the description of the FRAC_SBR bitfield.
FRAC_SBR	Fractional SCI Baud Rate divider, a value from 0 to 7 that is divided by 8 The SBR and FRAC_SBR fields combine to form the divider to determine the baud rate of the SCI. The integer portion of the baud rate divider is represented by SBR, and the fractional portion is represented by FRAC_SBR. The FRAC_SBR field can only be used when the integer portion is greater than 1. Therefore, the value of the divider can be 1.000 or (with fractional values) in the range from 2.000 to 8191.875. The formula for calculating the baud rate is: `Baud rate = peripheral bus clock / (16 * (SBR + (FRAC_SBR / 8)))` **NOTE:** The baud rate generator is disabled until CTRL1[TE] or CTRL1[RE] is set for the first time after reset. The baud rate generator is disabled when RATE[SBR] and RATE[FRAC_SBR] equal 0. **NOTE:** If CTRL2[LINMODE] is set, the value of this register is automatically adjusted to match the data rate of the LIN master device. Reading this register yields the auto-baud value set.

33.3.2 QSCI Control Register 1 (QSCIx_CTRL1)

Read: anytime

Write: anytime

Address: Base address + 1h offset

Bit	15	14	13	12	11	10	9	8
Read Write	LOOP	SWAI	RSRC	M	WAKE	POL	PE	PT
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
Read Write	TEIE	TIIE	RFIE	REIE	TE	RE	RWU	SBK
Reset	0	0	0	0	0	0	0	0

QSCIx_CTRL1 field descriptions

Field	Description
15 LOOP	Loop Select This bit enables loop operation. In loop operation the RXD pin is disconnected from the SCI, and the transmitter output goes into the receiver input. Both the transmitter and the receiver must be enabled to use the internal loop function as opposed to single wire operation, which requires only one or the other to be enabled. The receiver input is determined by CTRL1[RSRC]. The transmitter output is controlled by CTRL1[TE]. If CTRL1[TE] is set and CTRL1[LOOP]=1, the transmitter output appears on the TXD pin. If CTRL1[TE] is clear and CTRL1[LOOP]=1, the TXD pin is high-impedance. 0 Normal operation, regardless of the value of RSRC 1 When RSRC = 0: Loop mode with internal TXD fed back to RXD 1 When RSRC = 1: Single-wire mode with TXD output fed back to RXD
14 SWAI	Stop in Wait Mode This bit disables the SCI in wait mode 0 SCI enabled in wait mode 1 SCI disabled in wait mode
13 RSRC	Receiver Source When CTRL1[LOOP]=1, CTRL1[RSRC] determines the internal feedback path for the receiver. 0 Receiver input internally connected to transmitter output 1 Receiver input connected to TXD pin
12 M	Data Format Mode This bit determines whether data characters are eight or nine bits long. 0 One start bit, eight data bits, one stop bit 1 One start bit, nine data bits, one stop bit
11 WAKE	Wake-up Condition This bit determines which condition wakes the SCI: a one (address mark) in the most significant bit position of a received data character or an idle condition on the RXD pin. 0 Idle line wake-up 1 Address mark wake-up
10 POL	Polarity This bit determines whether to invert the data as it goes from the transmitter to the TXD pin and from the RXD pin to the receiver. All bits (start, data, and stop) are inverted as they leave the transmit shift register and before they enter the receive shift register.

Table continues on the next page...

QSCIx_CTRL1 field descriptions (continued)

Field	Description
	NOTE: It is recommended that CTRL1[POL] be toggled only when CTRL1[TE]=0 and CTRL1[RE]=0. 0 Don't invert transmit and receive data bits (normal mode) 1 Invert transmit and receive data bits (inverted mode)
9 PE	Parity Enable This bit enables the parity function. When enabled, the parity function replaces the most significant bit of the data character with a parity bit. 0 Parity function disabled 1 Parity function enabled
8 PT	Parity Type This bit determines whether the SCI generates and checks for even parity or odd parity of the data bits. With even parity, an even number of ones clears the parity bit and an odd number of ones sets the parity bit. With odd parity, an odd number of ones clears the parity bit and an even number of ones sets the parity bit. 0 Even parity 1 Odd parity
7 TEIE	Transmitter Empty Interrupt Enable This bit enables the transmit data register empty flag, STAT[TDRE], to generate interrupt requests. 0 STAT[TDRE] interrupt requests disabled 1 STAT[TDRE] interrupt requests enabled
6 TIIE	Transmitter Idle Interrupt Enable This bit enables the transmitter idle flag, STAT[TIDLE], to generate interrupt requests. 0 STAT[TIDLE] interrupt requests disabled 1 STAT[TIDLE] interrupt requests enabled
5 RFIE	Receiver Full Interrupt Enable This bit enables the receive data register full flag, STAT[RDRF], or the overrun flag, STAT[OR], to generate interrupt requests. 0 STAT[RDRF] and STAT[OR] interrupt requests disabled 1 STAT[RDRF] and STAT[OR] interrupt requests enabled
4 REIE	Receive Error Interrupt Enable This bit enables the receive error flags (STAT[NF], STAT[PF], STAT[FE], STAT[LSE], and STAT[OR]) to generate interrupt requests. 0 Error interrupt requests disabled 1 Error interrupt requests enabled
3 TE	Transmitter Enable This bit enables the SCI transmitter and configures the TXD pin as the SCI transmitter output. CTRL1[TE] can be used to queue an idle preamble. 0 Transmitter disabled 1 Transmitter enabled

Table continues on the next page...

QSCIx_CTRL1 field descriptions (continued)

Field	Description
2 RE	Receiver Enable This bit enables the SCI receiver. 0 Receiver disabled 1 Receiver enabled
1 RWU	Receiver Wake-up This bit enables the wake-up function and inhibits further receiver interrupt requests. Normally, hardware wakes the receiver by automatically clearing CTRL1[RWU]. 0 Normal operation 1 Standby state
0 SBK	Send Break Toggling this bit sends one break character (10 or 11 zeroes). As long as this bit is set, the transmitter sends zeroes. 0 No break characters 1 Transmit break characters

33.3.3 QSCI Control Register 2 (QSCIx_CTRL2)

Read: anytime

Write: anytime

Address: Base address + 2h offset

Bit	15	14	13	12	11	10	9	8
Read	TFCNT			TFWM		RFCNT		
Write								
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
Read	RFWM		FIFO_EN	RIEIE	LINMODE	0	TDE	RDE
Write								
Reset	0	0	0	0	0	0	0	0

QSCIx_CTRL2 field descriptions

Field	Description
15–13 TFCNT	Transmit FIFO Count These read only bits show how many words are used in the TX FIFO. Writes to DATA cause CTRL2[TFCNT] to increment. As words are pulled for transmission, CTRL2[TFCNT] decrements. Attempts to write new data to DATA are ignored when CTRL2[TFCNT] indicates the FIFO is full (4 words). 000 0 words in Tx FIFO

Table continues on the next page...

QSCIx_CTRL2 field descriptions (continued)

Field	Description
	001 1 word in Tx FIFO 010 2 words in Tx FIFO 011 3 words in Tx FIFO 100 4 words in Tx FIFO 101 Reserved 110 Reserved 111 Reserved
12–11 TFWM	Transmit FIFO Empty Water Mark These bits set the TX FIFO word count level at which STAT[TDRE] is set. If CTRL2[FIFO_EN] is clear (FIFO disabled), then this field is forced to 00 and STAT[TDRE] is set when there is no word in the transmit buffer. 00 TDRE is set when 0 words are in the FIFO 01 TDRE is set when 1 or fewer words are in the FIFO 10 TDRE is set when 2 or fewer words are in the FIFO 11 TDRE is set when 3 or fewer words are in the FIFO
10–8 RFCNT	Receive FIFO Count These read only bits show how many words are used in the RX FIFO. As words are received, CTRL2[RFCNT] is incremented. As words are read from DATA the value of CTRL2[RFCNT] decrements. There is one word time to read DATA between when STAT[RDRF] is set (interrupt asserted), due to the RX FIFO being full, and when an overflow condition is flagged. 000 0 words in RX FIFO 001 1 word in RX FIFO 010 2 words in RX FIFO 011 3 words in RX FIFO 100 4 words in RX FIFO 101 Reserved 110 Reserved 111 Reserved
7–6 RFWM	Receive FIFO Full Water Mark These bits set the RX FIFO word count level at which STAT[RDRF] is set. If CTRL2[FIFO_EN] is clear (FIFO disabled), then this field is forced to 00 and STAT[RDRF] is set if there is a word in the receive buffer. 00 RDRF is set when at least 1 word is in the FIFO 01 RDRF is set when at least 2 words are in the FIFO 10 RDRF is set when at least 3 words are in the FIFO 11 RDRF is set when at least 4 words are in the FIFO
5 FIFO_EN	FIFO Enable This read/write bit enables the 4-word-deep TX and RX FIFOs. Change this bit only when the SCI is idle. 0 FIFOs are disabled. 1 FIFOs are enabled.
4 RIEIE	Receiver Input Edge Interrupt Enable This bit is used to enable the receiver input active edge interrupt request with the STAT[RIEF] bit.

Table continues on the next page...

QSCIx_CTRL2 field descriptions (continued)

Field	Description
	0 Receiver input edge interrupt request disabled. 1 Receiver input edge interrupt request enabled.
3 LINMODE	Enable LIN Slave Mode This bit should be used only in Local Interconnect Network (LIN) applications. **NOTE:** During initialization, the RATE register should be loaded to a value that is within 14% of the actual master data rate; otherwise, 0x00 data might be misinterpreted as a break. **NOTE:** If the first character following a break is not the LIN sync character (0x55), the RATE register is not adjusted and STAT[LSE] is set. 0 The LIN auto baud feature is disabled and the RATE register maintains whatever value the processor writes to it. 1 Enable LIN slave functionality. This includes a search for the break character followed by a sync character (0x55) from the master LIN device. When the break is detected (11 consecutive samples of zero), the subsequent sync character is used to measure the baud rate of the transmitting master, and the RATE register is automatically reloaded with the value needed to "match" that baud rate.
2 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1 TDE	Transmitter DMA Enable This bit is used to enable DMA mode transfer of data to the DATA register. For transmit DMA operation this bit must be set and a DMA channel must be enabled to use the request. 0 Transmit DMA disabled 1 Transmit DMA enabled
0 RDE	Receiver DMA Enable This bit is used to enable DMA mode transfer of data from the DATA register. For receive DMA operation this bit must be set and a DMA channel must be enabled to utilize the request. 0 Receive DMA disabled 1 Receive DMA enabled

33.3.4 QSCI Status Register (QSCIx_STAT)

Read: anytime

Write: this register is not writable

Address: Base address + 3h offset

Bit	15	14	13	12	11	10	9	8
Read	TDRE	TIDLE	RDRF	RIDLE	OR	NF	FE	PF
Write								
Reset	1	1	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
Read	0			RIEF	LSE	TDMA	RDMA	RAF
Write								
Reset	0	0	0	0	0	0	0	0

QSCIx_STAT field descriptions

Field	Description
15 TDRE	Transmit Data Register Empty Flag This bit is set when the TX FIFO word count (CTRL2[TFCNT]) falls to the watermark level (CTRL2[TXWM]). Clear TDRE by reading STAT with TDRE set and then writing to the SCI data register until CTRL2[TFCNT] is above CTRL2[TFWM]. If CTRL2[FIFO_EN] or CTRL2[TDE] is set, then you can clear TDRE by writing to the SCI data register without first reading STAT with TDRE set. 0 TX FIFO word count is above watermark 1 TX FIFO word count is at or below watermark
14 TIDLE	Transmitter Idle Flag This bit is set when the TX FIFO is empty and no data, preamble, or break character is being transmitted. When TIDLE is set, the TXD pin becomes idle (1). Clear TIDLE by reading STAT with TIDLE set and then writing to the SCI data register. TIDLE is not generated when a data character, a preamble, or a break is queued and ready to be sent. 0 Transmission in progress 1 No transmission in progress
13 RDRF	Receive Data Register Full Flag This bit is set when the RX FIFO word count (CTRL2[RFCNT]) rises above the watermark (CTRL2[RXWM]). Clear RDRF by reading STAT with RDRF set and then reading the SCI data register until CTRL2[RFCNT] is no longer above CTRL2[RFWM]. If CTRL2[FIFO_EN] or CTRL2[RDE] is set, then you can clear RDRF by reading the SCI data register without first reading STAT with RDRF set. **NOTE:** When you are using the CodeWarrior debugger, RDRF may be erased when a breakpoint is reached. If a memory window that includes the SCI registers is open when a breakpoint is reached, these memory addresses are read to update the memory window. If RDRF is set at this time, these reads satisfy the requirements for clearing RDRF because the status register is read with RDRF set and then the data register is read, which causes RDRF to clear. 0 RX FIFO word count is at or below watermark 1 RX FIFO word count is above watermark
12 RIDLE	Receiver Idle Line Flag This bit is set when 10 consecutive ones (if CTRL1[M]=0) or 11 consecutive ones (if CTRL1[M]=1) appear on the receiver input. The RIDLE flag is cleared by reading STAT with RIDLE set and then reading the SCI data register. Once RIDLE is cleared, a valid frame must be received before an idle condition can set RIDLE. When the receiver wake-up bit (CTRL1[RWU]) is 0, the user can poll RIDLE to detect when the receiver is idle. **NOTE:** When the receiver wake-up bit (CTRL1[RWU]) is set to 1, an idle line condition does not set RIDLE. 0 Receiver input is either active now or has never become active since RIDLE was last cleared 1 Receiver input has become idle (after receiving a valid frame)

Table continues on the next page...

QSCIx_STAT field descriptions (continued)

Field	Description
11 OR	Overrun Flag This bit is set when software fails to read the SCI data register when the RX FIFO is full before the receive shift register receives the next frame. The data in the shift register is lost, but the data already in the SCI data register/FIFO is not affected. Clear OR by reading STAT with OR set and then writing the SCI status register with any value. **NOTE:** When the Overrun Flag is set and remains set, other flags cannot become set. The receive FIFO is full and cannot receive any more words, so the arriving data is ignored and cannot set any other flags. 0 No overrun 1 Overrun
10 NF	Noise Flag This bit is set when the SCI detects noise on the receiver input. NF is set during the same cycle as RDRF but is not set in the case of an overrun. Clear NF by reading STAT and then writing the SCI status register with any value. 0 No noise 1 Noise
9 FE	Framing Error Flag This bit is set when a 0 is accepted as the stop bit. FE is set during the same cycle as RDRF but is not set in the case of an overrun. Clear FE by reading STAT with FE set and then writing the SCI status register with any value. 0 No framing error 1 Framing error
8 PF	Parity Error Flag This bit is set when the parity enable bit (CTRL1[PE]) is set and the parity of the received data does not match its parity bit. Clear PF by reading STAT and then writing the SCI status register with any value. 0 No parity error 1 Parity error
7–5 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
4 RIEF	Receiver Input Edge Flag This flag is set when an active edge is seen on the RXD input pin. An active edge is defined as a 1 to 0 transition when CTRL[POL] is 0 or as a 0 to 1 transition when CTRL[POL] is 1. This flag can be used in stop mode (clocks turned off) to create an interrupt to wake the CPU when the leading edge of a start bit is detected. As to whether the SOC can start up clocking fast enough for the SCI to properly detect the start bit and receive the data word is a function of SOC architecture and SCI baud rate. Clear RIEF by writing a 1 to this bit position. 0 No active edge on the receive pin has occured. 1 An active edge on the receive pin has occured.
3 LSE	LIN Sync Error This bit is active only when CTRL2[LINMODE] is set. When LSE is set, a Receive Error interrupt occurs if CTRL1[REIE] is set.

Table continues on the next page...

QSCIx_STAT field descriptions (continued)

Field	Description
	LSE is set when a LIN sync search detects a non-sync character (anything other than 0x55). When set, this bit indicates either that a protocol error was detected from the Master LIN device or there is a gross mismatch in data rates. This bit is cleared by reading STAT with LSE set and then writing the SCI status register with any value. 0 No error occurred since CTRL2[LINMODE] was enabled or the bit was last cleared 1 A sync error prevented loading of the RATE register with a revised value after the break was detected.
2 TDMA	Transmit DMA Request When set, this bit indicates that the SCI is currently requesting a DMA data transfer for transmit data. This bit is cleared when the DMA channel writes enough data to the DATA register to raise CTRL2[TFCNT] to its maximum value. 0 Either CTRL2[TDE] is cleared or CTRL2[TDE] is set and CTRL2[TFCNT] is at its maximum value. 1 CTRL2[TDE] is set and CTRL2[TFCNT] is currently below its maximum value.
1 RDMA	Receive DMA Request When set, this bit indicates that the SCI is currently requesting a DMA data transfer for received data. This bit is cleared when the DMA channel reads enough data from the DATA register to lower CTRL2[RFCNT] to 0. 0 Either CTRL2[RDE] is cleared or CTRL2[RDE] is set and CTRL2[RFCNT] is 0. 1 CTRL2[RDE] is set and CTRL2[RFCNT] is currently above 0.
0 RAF	Receiver Active Flag This bit is set when the receiver detects a 0 during the RT1 time period of the start bit search. RAF is cleared when the receiver detects false start bits (usually from noise or baud rate mismatch) or when the receiver detects a preamble. 0 No reception in progress 1 Reception in progress

33.3.5 QSCI Data Register (QSCIx_DATA)

Read: anytime. Reading accesses the SCI receive data register.

Write: anytime. Writing accesses the SCI transmit data register.

Address: Base address + 4h offset

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	0								RECEIVE_TRANSMIT_DATA							
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

QSCIx_DATA field descriptions

Field	Description
15–9 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

Table continues on the next page...

QSCIx_DATA field descriptions (continued)

Field	Description
RECEIVE_ TRANSMIT_ DATA	RECEIVE_DATA: Received data (when reading this register) TRANSMIT_DATA: Data to be transmitted (when writing this register) If STAT[TDRE] is set, writes to the transmit data field are ignored unless STAT has been read with STAT[TDRE] set. However, when CTRL2[FIFO_EN] or CTRL2[TDE] are set, you can write to the SCI data register without first reading STAT with STAT[TDRE] set. If STAT[RDRF] is set, reads from the receive data field are ignored unless STAT has been read with STAT[RDRF] set. However, when CTRL2[FIFO_EN] or CTRL2[RDE] are set, you can read from the SCI data register without first reading STAT with STAT[RDRF] set. This register is a 4-deep FIFO. **NOTE:** When the data format is configured for 8-bit data, bits 7 to 0 contain the data.

33.3.6 QSCI Control Register 3 (QSCIx_CTRL3)

Read: anytime.

Write: anytime.

Address: Base address + 5h offset

Bit	15	14	13	12	11	10	9	8
Read	0							
Write								
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
Read	0							SHEN
Write								
Reset	0	0	0	0	0	0	0	0

QSCIx_CTRL3 field descriptions

Field	Description
15–1 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
0 SHEN	Stop mode entry hold off Setting this bit allows the SCI to hold off chip level stop mode entry while the transmitter is not idle or while the transmit data register is not empty. It will also hold off stop mode entry while the receiver is not idle. Clearing this bit means that stop mode entry is not delyed and may occur while the SCI is actively transmitting/receiving. Under no cases will this bit affect wake up from stop mode once stop mode has been entered. **NOTE:** Not all chips support stop mode hold off. 0 Stop mode hold off is disabled. 1 Stop mode holdoff is enabled.

33.4 Functional Description

The SCI block diagram shows the structure of the SCI module. The SCI allows full duplex, asynchronous, NRZ serial communication between the DSP and remote devices, including other DSPs. The SCI transmitter and receiver operate independently, although they use the same baud rate generator. The core monitors the status of the SCI, writes the data to be transmitted, and processes received data.

When initializing the SCI, be sure to set the proper peripheral enable bits in the GPIO as well as any pullup enables.

33.4.1 Data Frame Format

The SCI uses the standard NRZ mark/space data frame format illustrated in the following figure.

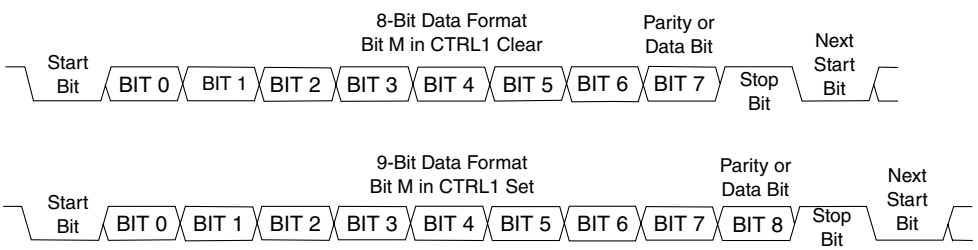


Figure 33-2. SCI Data Frame Formats

Each data character is contained in a frame that includes a start bit, eight or nine data bits, and a stop bit. Clearing CTRL1[M] configures the SCI for 8-bit data characters. A frame with eight data bits has a total of 10 bits, with formats as shown in the following table.

Table 33-2. Example 8-Bit Data Frame Formats

Start Bit	Data Bits	Address Bit	Parity Bit	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.

Setting CTRL1[M] configures the SCI for 9-bit data characters. A frame with nine data bits has a total of 11 bits with formats as shown in the following table.

Table 33-3. Example 9-Bit Data Frame Formats

Start Bit	Data Bits	Address Bit	Parity Bit	Stop Bit
1	9	0	0	1
1	8	0	0	2
1	8	0	1	1
1	8	1[1]	0	1

1. The address bit identifies the frame as an address character.

33.4.2 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 written to the RATE[SBR] and RATE[FRAC_SBR] bits determines the peripheral bus clock divisor. The baud rate clock is synchronized with the IP 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 two sources of error:

- Integer division of the peripheral bus clock may not give the exact target frequency.
- Synchronization with the bus clock can cause phase shift.

The following table lists some examples of achieving target baud rates with a peripheral bus clock frequency of 60 MHz.

Table 33-4. Example Baud Rates (Peripheral Bus Clock = 60 MHz)

SBR Bits	Receiver Clock (Hz)	Transmitter Clock (Hz)	Target Baud Rate	Error (%)
32.5	1,846,154	115,385	115,200	0.16
65.125	921,305	57,582	57,600	-0.03
97.625	614,597	38,412	38,400	0.03
195.25	307,298	19,206	19,200	0.03
390.625	153,600	9,600	9600	0.00
781.25	76,800	4,800	4800	0.00
1562.5	38,400	2,400	2400	0.00
3125	19,200	1,200.0	1200	0.00
6250	9,600	600.0	600	0.00

The following table lists some examples of achieving target baud rates with a peripheral bus clock frequency of 32 MHz.

Table 33-5. Example Baud Rates (Peripheral Bus Clock = 32 MHz)

SBR Bits	Receiver Clock (Hz)	Transmitter Clock (Hz)	Target Baud Rate	Error (%)
17.375	1,841,727	115,108	115,200	-0.08
34.75	920,863	57,554	57,600	-0.08
52.125	613,909	38,369	38,400	-0.08
104.125	307,323	19,208	19,200	0.04
208.375	153,569	9,598	9,600	-0.02
416.625	76,808	4,800	4,800	0.01
833.375	38,398	2,400	2,400	-0.01
1666.625	19,200	1,200	1,200	0.00
3333.375	9,600	600	600	0.00

Note

Maximum baud rate is peripheral bus clock rate divided by 16. System overhead may preclude processing the data at this speed.

33.4.3 Transmitter

The following figure is a block diagram of the transmitter functions.

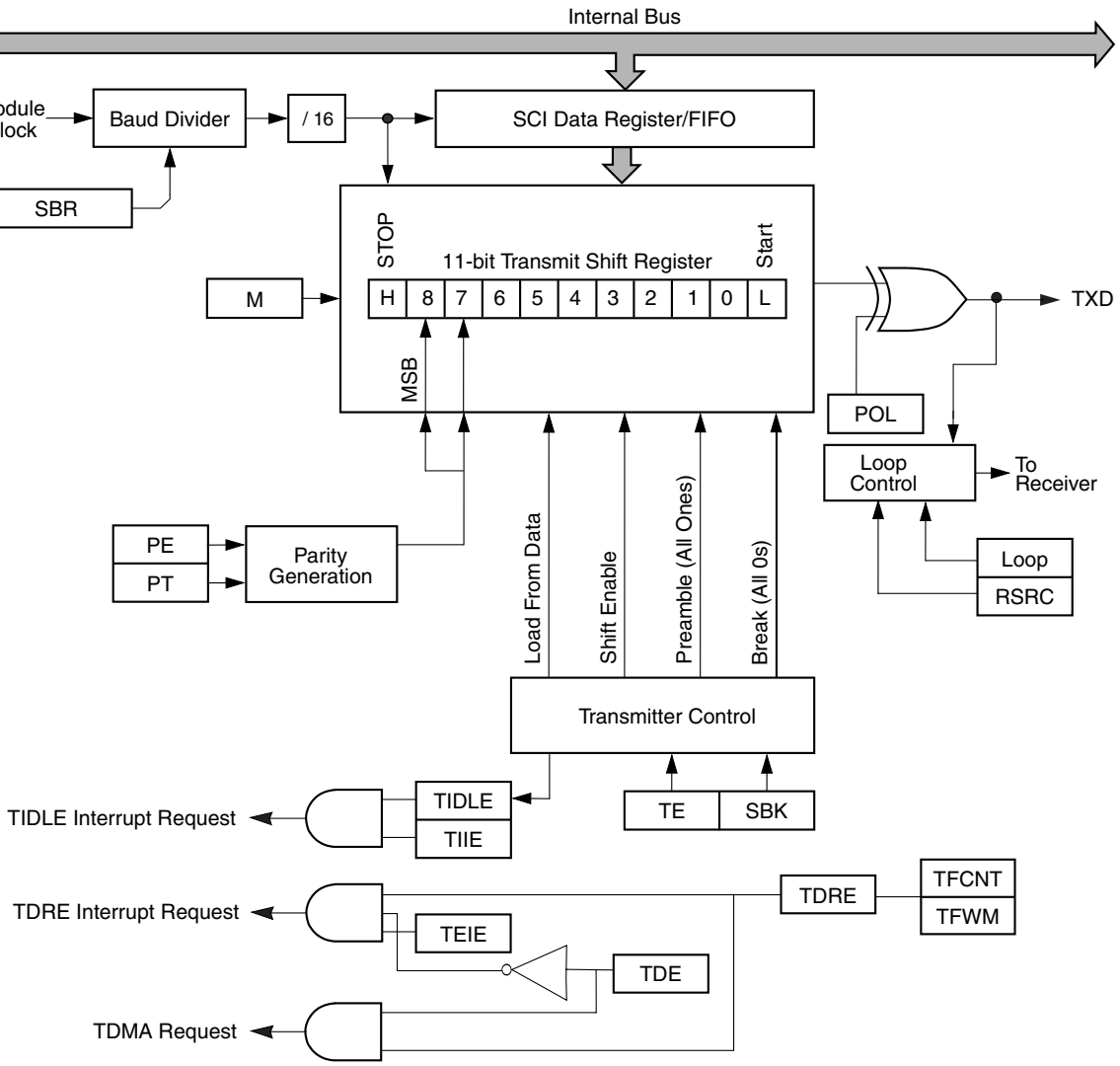


Figure 33-3. SCI Transmitter Block Diagram with DMA

33.4.3.1 Character Length

The SCI transmitter can accommodate either 8-bit or 9-bit data characters. The state of CTRL1[M] determines the length of data characters.

33.4.3.2 Character Transmission

During an SCI transmission, the transmit shift register shifts a frame out to the TXD pin. The SCI data register is the write-only buffer between the internal data bus and the transmit shift register.

To initiate an SCI transmission:

1. Enable the transmitter by writing a logic 1 to the transmitter enable bit (CTRL1[TE]).
2. Clear the transmit data register empty flag, STAT[TDRE], by first reading the SCI status register (STAT) and then writing to the SCI data register (DATA).
3. Repeat step 2 for each subsequent transmission.

Writing CTRL1[TE] bit from 0 to a 1 automatically loads the transmit shift register with a preamble of 10 ones (if CTRL1[M] = 0) or 11 ones (if CTRL1[M] = 1). After the preamble shifts out, control logic automatically transfers the data from the SCI data register into the transmit shift register. A logic zero start bit automatically goes into the least significant bit position of the transmit shift register. A logic one stop bit goes into the most significant bit (MSB) position of the frame.

Hardware supports odd or even parity. When parity is enabled, the MSB of the data character is replaced by the parity bit.

The transmit data register empty flag, STAT[TDRE], becomes set when the SCI data register transfers a character to the transmit shift register. STAT[TDRE] indicates that the SCI data register can accept new data from the internal data bus. If the transmitter empty interrupt enable bit, CTRL1[TEIE], is also set, STAT[TDRE] generates a transmitter interrupt request.

When the transmit shift register is not transmitting a frame and CTRL1[TE]=1, the TXD pin goes to the idle condition, logic one. If at any time software clears CTRL1[TE], the transmitter relinquishes control of the port I/O pin upon completion of the current transmission, causing the TXD pin to go to a high z state.

If software clears CTRL1[TE] while a transmission is in progress (STAT[TIDLE] = 0), the frame in the transmit shift register continues to shift out. Then transmission stops even if there is data pending in the SCI data register. To avoid accidentally cutting off the last frame in a message, always wait for STAT[TDRE] to go high after the last frame before clearing CTRL1[TE].

To separate messages with preambles with minimum idle line time, use this sequence between messages:

1. Write the last character of the first message to the DATA register.
2. Wait for STAT[TDRE] to go high while CTRL2[TFWM] = 00, indicating the transfer of the last frame to the transmit shift register.
3. Queue a preamble by clearing and then setting CTRL1[TE].
4. Write the first character of the second message to the DATA register.

33.4.3.3 Break Characters

Writing a logic one to the send break bit, CTRL1[SBK], loads the transmit shift register with a break character. A break character contains all logic zeroes and has no start, stop, or parity bit. Break character length depends on CTRL1[M]. As long as CTRL1[SBK] is at logic one, transmitter logic continuously loads break characters into the transmit shift register. After software clears CTRL1[SBK], the shift register finishes transmitting the last break character and then transmits at least one logic one. The automatic logic one at the end of the last break character guarantees the recognition of the start bit of the next frame.

In order to send an 11 bit break character, set the CTRL1[M] bit, set and then clear CTRL1[SBK], then poll STAT[TIDLE]. Once STAT[TIDLE] is asserted the CTRL1[M] bit can be cleared in order for future data words to be 8 bits long.

The SCI recognizes a break character when a start bit is followed by eight or nine logic zero data bits and a logic zero where the stop bit should be. Receiving a break character has these effects on SCI registers:

- Sets the framing error flag, STAT[FE]
- Sets the receive data register full flag, STAT[RDRF], if CTRL2[RFWM] = 00
- Clears the SCI data register
- May set the overrun flag (STAT[OR]), noise flag (STAT[NF]), parity error flag (STAT[PF]), or receiver active flag (STAT[RAF])

33.4.3.4 Preambles

A preamble contains all logic ones and has no start, stop, or parity bit. Preamble length depends on CTRL1[M]. The preamble is a synchronizing mechanism that begins the first transmission initiated after writing CTRL1[TE] from 0 to 1.

If CTRL1[TE] is cleared during a transmission, the TXD pin becomes idle after completion of the transmission in progress. Clearing and then setting CTRL1[TE] during a transmission queues a preamble to be sent after the frame currently being transmitted.

Note

Toggle CTRL1[TE] for a queued preamble when STAT[TDRE] becomes set and immediately before writing the next character to the DATA register.

When queueing a preamble, return CTRL1[TE] to logic one before the stop bit of the current frame shifts out to the TXD pin. Setting CTRL1[TE] after the stop bit appears on TXD causes data previously written to the SCI data register to be lost.

33.4.4 Receiver

The following figure is the block diagram of the SCI receiver.

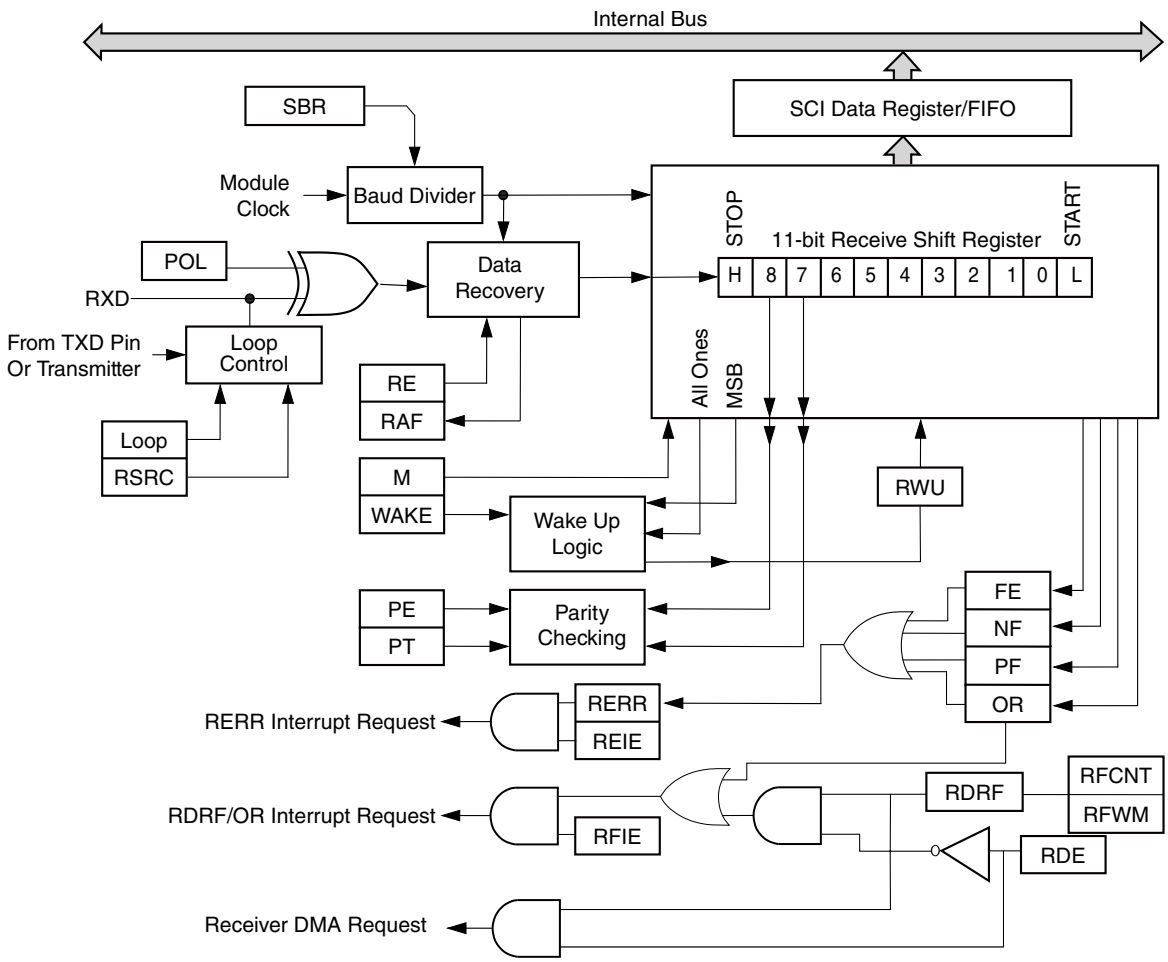


Figure 33-4. SCI Receiver Block Diagram with DMA

33.4.4.1 Character Length

The SCI receiver can accommodate either 8-bit or 9-bit data characters. The state of CTRL1[M] determines the length of data characters.

33.4.4.2 Character Reception

During an SCI reception, the receive shift register shifts a frame in from the RXD pin. The SCI data register/FIFO 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 along with the STAT[FE], STAT[NF], STAT[PF], and STAT[LSE] status flags transfer to the SCI data register. The receive data register full flag, STAT[RDRF], becomes set when the RX FIFO word count is above the watermark, indicating that a received character can be read. When the FIFO is enabled, there can be received data words to be read even if STAT[RDRF] is not set. If the receive interrupt enable bit, CTRL1[RFIE], is also set, STAT[RDRF] generates a Receiver Full interrupt request.

The STAT[FE], STAT[NF], STAT[PF], and STAT[LSE] flags are associated with the current character to be read from the receive data register/FIFO. When responding to a receiver error interrupt, read both the DATA register and the STAT register, and then write the STAT register to clear the error flags. When responding to a receiver full interrupt, it is necessary to read only the DATA register.

33.4.4.3 Data Sampling

The receiver samples the RXD pin at 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 (shown in the following figure) is resynchronized:

- After every start bit
- After the receiver detects a data bit change from logic 1 to logic 0 (after most data bit samples at RT8, RT9, and RT10 return a valid logic 1 and most of the next RT8, RT9, and RT10 samples return a valid logic 0)

To locate the start bit, data recovery logic does an asynchronous search for a logic 0 preceded by three logic ones. When the falling edge of a possible start bit occurs, the RT clock begins to count to 16.

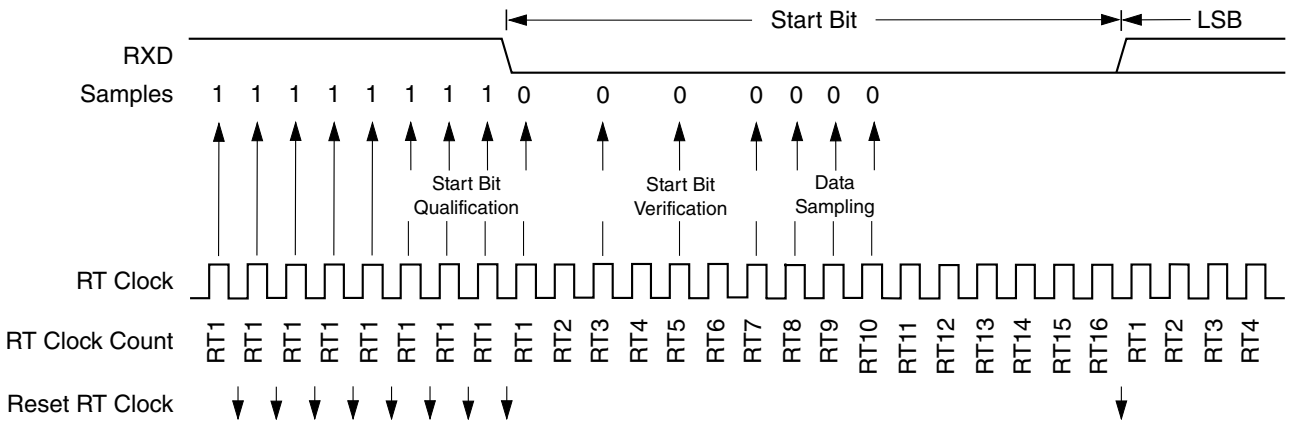


Figure 33-5. Receiver Data Sampling

To verify the start bit and to detect noise, data recovery logic takes samples at RT3, RT5, and RT7. The following table summarizes the results of the start bit verification samples.

Table 33-6. 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. The following table summarizes the results of the data bit samples.

Table 33-7. 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 ones following a successful start bit verification, the noise flag (STAT[NF]) is set and the receiver assumes that the bit is a start bit (logic zero).

To verify a stop bit and to detect noise, recovery logic takes samples at RT8, RT9, and RT10. The following table summarizes the results of the stop bit samples.

Table 33-8. 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 the following figure, 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.

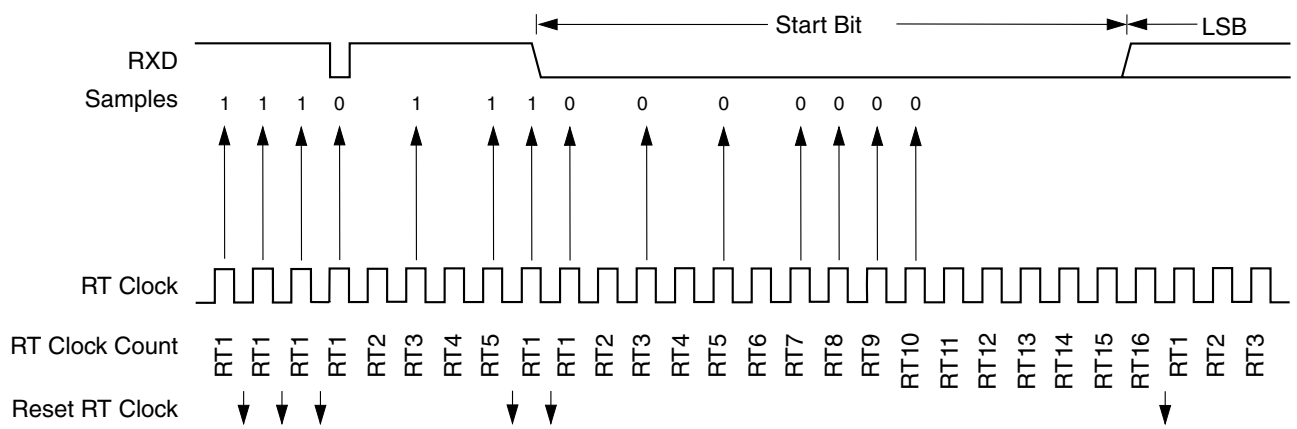


Figure 33-6. Start Bit Search Example 1

In the following figure, noise is perceived as the beginning of a start bit because the 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.

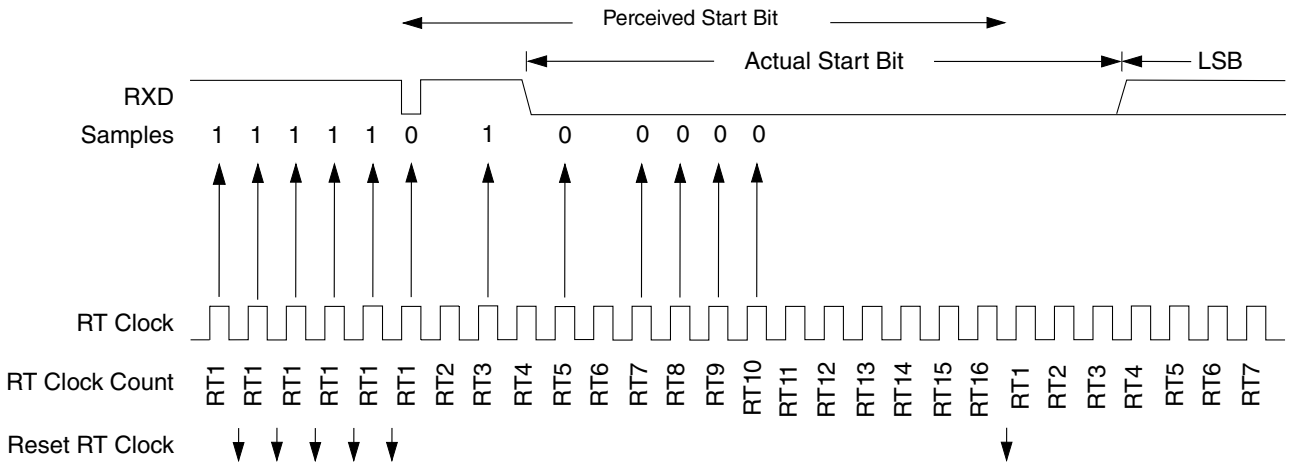


Figure 33-7. Start Bit Search Example 2

In the following figure, 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.

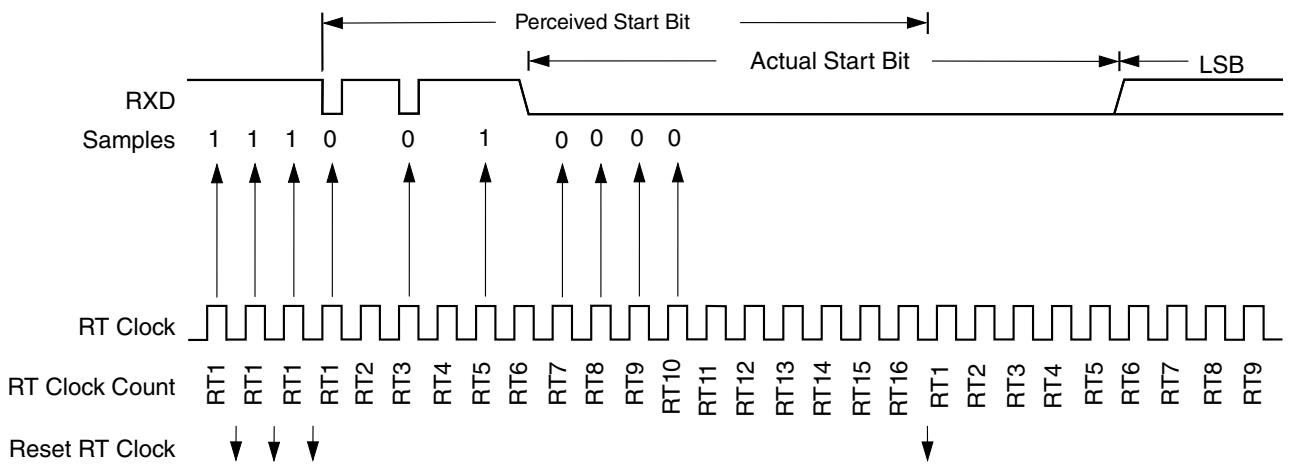


Figure 33-8. Start Bit Search Example 3

The following figure 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.

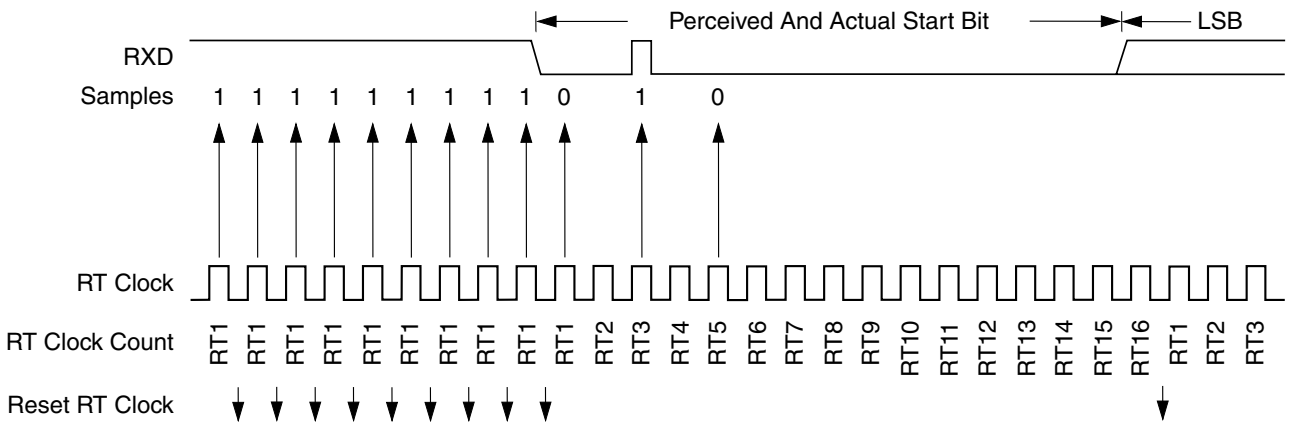


Figure 33-9. Start Bit Search Example 4

The following figure 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.

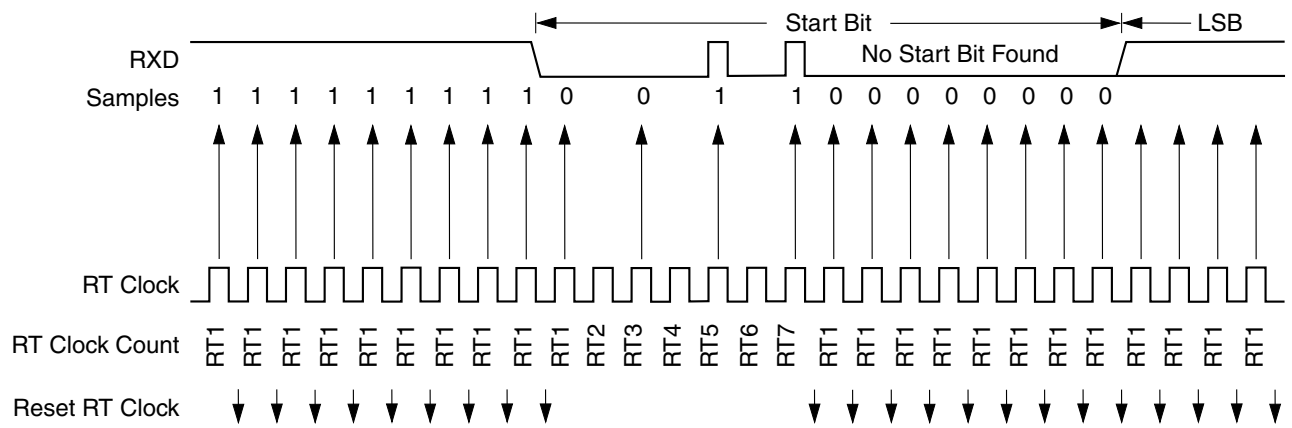


Figure 33-10. Start Bit Search Example 5

In the following figure, 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.

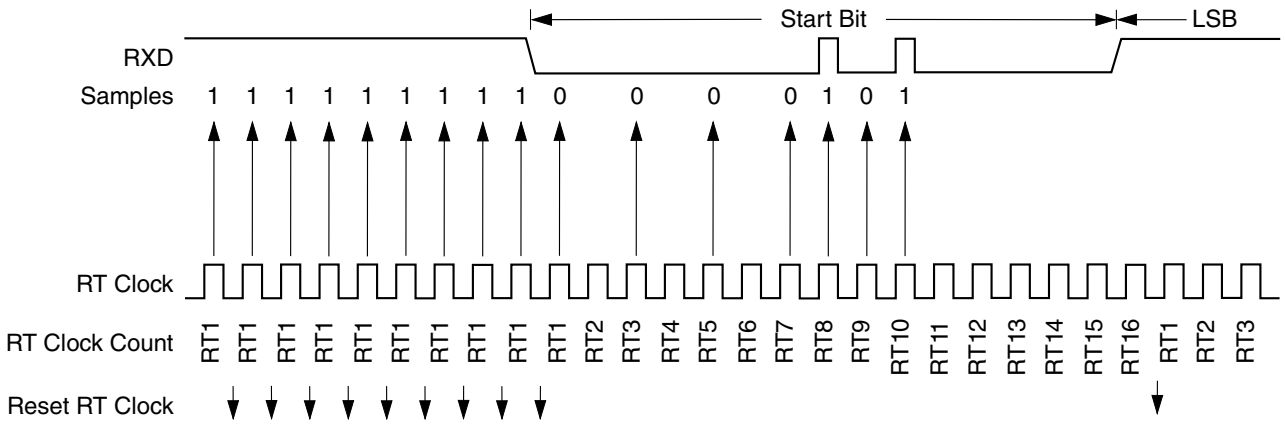


Figure 33-11. Start Bit Search Example 6

33.4.4.4 Framing Errors

If the data recovery logic does not detect a logic one where the stop bit should be in an incoming frame, it sets the framing error flag, STAT[FE]. A break character also sets STAT[FE] because a break character has no stop bit. STAT[FE] is set at the same time that STAT[RDRF] is set.

33.4.4.5 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 to fall outside the actual stop bit. Then a noise error occurs. If more than one of the samples is outside the stop bit, a framing error occurs. In most applications, the baud rate tolerance is much more than the degree of misalignment that is likely to occur.

As the receiver samples an incoming frame, it resynchronizes the RT clock on any valid falling edge within the frame. Resynchronization within frames corrects misalignments between transmitter bit times and receiver bit times.

33.4.4.6 Slow Data Tolerance

The following figure 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.

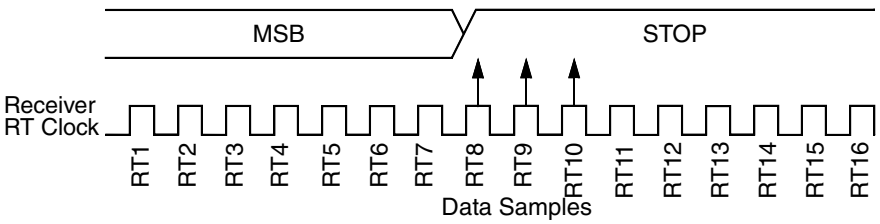


Figure 33-12. Slow Data

For an 8-bit data character, data sampling of the stop bit takes the receiver the following number of cycles:

```
9 bit × 16 RT cycles + 10 RT cycles = 154 RT cycles
```

With the misaligned character shown in the data figure, the receiver counts 154 RT cycles at the point when the count of the transmitting device is:

```
9 bit × 16 RT cycles + 3 RT cycles = 147 RT cycles
```

The maximum percent difference between the receiver count and the transmitter count of a slow 8-bit data character with no errors is:

```
((154-147) / 154) = 4.54%
```

For a 9-bit data character, data sampling of the stop bit takes the receiver the following number of cycles:

```
10 bit × 16 RT cycles + 10 RT cycles = 170 RT cycles
```

With the misaligned character shown in the slow data figure, the receiver counts 170 RT cycles at the point when the count of the transmitting device is:

```
10 bit × 16 RT cycles + 3 RT cycles = 163 RT cycles
```

The maximum percent difference between the receiver count and the transmitter count of a slow 9-bit character with no errors is:

```
((170-163) / 170) = 4.12%
```

33.4.4.7 Fast Data Tolerance

The following figure shows how much a fast received frame can be misaligned without causing a noise error or a framing error. The fast stop bit ends at RT10 instead of RT16 but is still sampled at RT8, RT9, and RT10.

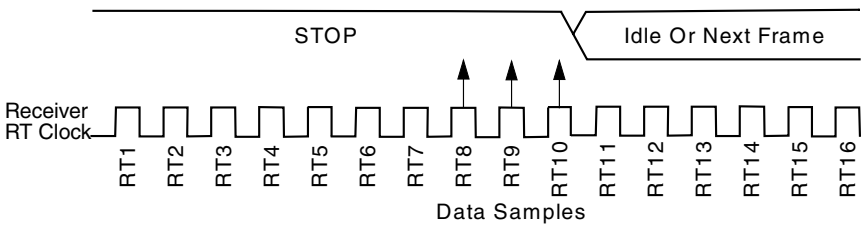


Figure 33-13. Fast Data

For an 8-bit data character, data sampling of the stop bit takes the receiver the following number of cycles:

```
9 bit × 16 RT cycles + 10 RT cycles = 154 RT cycles
```

With the misaligned character shown in the fast data figure, the receiver counts 154 RT cycles at the point when the count of the transmitting device is:

```
10 bit × 16 RT cycles = 160 RT cycles
```

The maximum percent difference between the receiver count and the transmitter count of a fast 8-bit character with no errors is:

```
((154-160) / 154) = 3.90%
```

For a 9-bit data character, data sampling of the stop bit takes the receiver the following number of cycles:

```
10 bit × 16 RT cycles + 10 RT cycles = 170 RT cycles
```

With the misaligned character shown in the fast data figure, the receiver counts 170 RT cycles at the point when the count of the transmitting device is:

```
11 bit × 16 RT cycles = 176 RT cycles
```

The maximum percent difference between the receiver count and the transmitter count of a fast 9-bit character with no errors is:

```
((170-176) / 170) = 3.53%
```

33.4.4.8 Receiver Wakeup

So that the SCI can 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, CTRL1[RWU], puts the receiver into a standby state during which receiver interrupts are disabled.

The transmitting device can address messages to selected receivers by including addressing information in the initial frame or frames of each message.

CTRL1[WAKE] determines how the SCI is brought out of the standby state to process an incoming message. CTRL1[WAKE] enables either idle line wakeup or address mark wakeup:

- Idle input line wakeup (CTRL1[WAKE] = 0): In this wakeup method, an idle condition on the RXD pin clears CTRL1[RWU] and wakes the SCI. The initial frame (or frames) of every message contains 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 CTRL1[RWU] bit and return to the standby state. CTRL1[RWU] remains set and the receiver remains on standby until another preamble appears on the RXD pin.

  The idle line wakeup method requires that messages be separated by at least one preamble and that no message contains preambles.

  The preamble that wakes a receiver does not set the receiver idle bit, STAT[RIDLE], or the receive data register full flag, STAT[RDRF].

- Address mark wakeup (CTRL1[WAKE] = 1): In this wakeup method, a logic one in the most significant bit (MSB) position of a frame clears CTRL1[RWU] and wakes up the SCI. The logic one 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. CTRL1[RWU] remains set and the receiver remains on standby until another address frame appears on the RXD pin.

  The logic one MSB of an address frame clears the receiver's CTRL1[RWU] bit before the stop bit is received and sets STAT[RDRF].

  The address mark wakeup method allows messages to contain preambles but requires that the MSB be reserved for use in address frames.

Note

With CTRL1[WAKE] clear, setting CTRL1[RWU] after the RXD pin is idle can cause the receiver to wake up immediately.

33.4.4.9 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 but is enabled, meaning CTRL1[RE] must be 1, and is available as a general-purpose I/O pin. The SCI uses the TXD pin for both receiving and transmitting.

Setting CTRL1[TE] configures TXD as the output for transmitted data. Clearing CTRL1[TE] configures TXD as the input for received data.

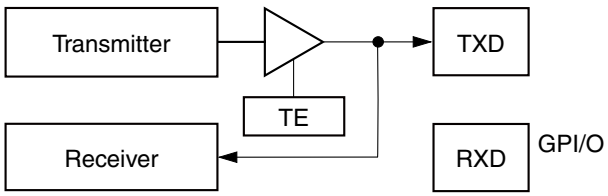


Figure 33-14. Single-Wire Operation (CTRL1[LOOP] = 1, CTRL1[RSRC] = 1)

Enable single-wire operation by setting CTRL1[LOOP] and the receiver source bit, CTRL1[RSRC]. Setting CTRL1[LOOP] disables the path from the RXD pin to the receiver. Setting CTRL1[RSRC] connects the receiver input to the output of the TXD pin driver.

33.4.4.10 Loop Operation

In loop operation the transmitter output goes to the receiver input. The RXD pin is disconnected from the SCI and is available as a general-purpose I/O pin.

Setting CTRL1[TE] connects the transmitter output to the TXD pin. Clearing CTRL1[TE] disconnects the transmitter output from the TXD pin.

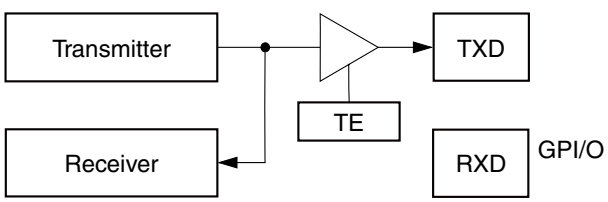


Figure 33-15. Loop Operation (CTRL1[LOOP] = 1, CTRL1[RSRC] = 0)

Enable loop operation by setting CTRL1[LOOP] and clearing CTRL1[RSRC]. Setting CTRL1[LOOP] disables the path from the RXD pin to the receiver. Clearing CTRL1[RSRC] connects the transmitter output to the receiver input. Both the transmitter and receiver must be enabled (CTRL1[TE] = 1 and CTRL1[RE] = 1).

33.4.5 DMA Operation

DMA operation is an optional feature that may not be implemented on all chips.

33.4.5.1 Transmit DMA Operation

Setting CTRL2[TDE] enables Transmit DMA mode. In this mode, the transmitter empty interrupt is suppressed, and instead a transmitter DMA request is generated whenever there is an empty space in the TX FIFO. The DMA controller then writes to the DATA register, clearing the request.

33.4.5.2 Receive DMA Operation

Setting CTRL2[RDE] enables Receiver DMA mode. In this mode, the Receiver Full interrupt is suppressed, and instead a Receiver DMA request is generated whenever there is data in the RX FIFO. The DMA controller then reads the DATA register, clearing the request.

33.4.5.3 Receiver Wakeup with DMA

If DMA operation is desired during either of the wakeup modes of operation, DMA requests should only be made for a message that is addressed to the receiver. In other words, wakeup operation proceeds normally until a desired receive message is identified. Then DMA requests are generated to buffer the remainder of the message. An example of the DMA receive operations is shown in Table 33-9.

Table 33-9. Receive DMA Operations

1. Configure the SCI for standard receive operation 2. SCI receives the first frame of the incoming message and stores it in the DATA register 3. A SCI Receiver Full interrupt occurs 4. The ISR looks at the message address information and determines that:	
MESSAGE NOT FOR US	**MESSAGE IS FOR US**
5. Configure SCI for RWU mode and wait for the end of the message. Repeat from step 1.	5. Enable DMA operation 6. Enable DMA interrupts at the completion of the message. 7. When the DMA interrupt occurs: process the message and return to step 1.

33.4.6 LIN Slave Operation

LIN slave operation occurs when CTRL2[LIN MODE] is set. The receiver searches for a break character consisting of at least 11 consecutive samples of logic zero, the next field to be received is the sync field. The sync field is a word with 0x55 data that produces an alternating 0 and 1 pattern. The receiver detects the falling edge at the beginning of the

start bit and starts counting system clocks until the falling edge at the beginning of data bit 7 is detected, at which point it stops counting. This count is divided by 8 (for the 8-bit periods that have passed) and further divided by 16 to provide new RATE[SBR] and RATE[FRAC_SBR] values. If the data value of the sync field is 0x55, then these new RATE[SBR] and RATE[FRAC_SBR] values are placed in the baud rate register. Then the slave is considered synced to the master and further data words are received properly. If the data value of the sync field isn't 0x55, then the LIN sync error (STAT[LSE]) bit is set and subsequent received data bytes should be ignored.

To detect the break character successfully, the initial baud rate for this slave device must be within 15 percent of the nominal baud rate for the LIN master device.

33.4.7 Low-Power Options

33.4.7.1 Run Mode

Clearing the transmitter enable or receiver enable bits (CTRL1[TE] or CTRL1[RE]) reduces power consumption in run mode. SCI registers are still accessible when CTRL1[TE] or CTRL1[RE] is cleared, but clocks to the core of the SCI are disabled.

33.4.7.2 Wait Mode

SCI operation in wait mode depends on the state of CTRL1[SWAI].

- If CTRL1[SWAI] is clear, the SCI operates normally when the core is in wait mode.
- If CTRL1[SWAI] is set, SCI clock generation ceases and the SCI module enters a power-conservation state when the core is in wait mode. SCI registers are not accessible. Setting CTRL1[SWAI] does not affect the state of the receiver enable bit, CTRL1[RE], or the transmitter enable bit, CTRL1[TE].

  If CTRL1[SWAI] 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 core out of wait mode. Exiting wait mode via reset aborts any transmission or reception in progress and resets the SCI.

33.4.7.3 Stop Mode

SCI operation in stop mode depends on the state of the SCI stop disable bit in the SIM's applicable stop disable register.

- If the SCI stop disable bit is clear, the SCI is inactive in stop mode to reduce power consumption. The STOP instruction does not affect the SCI registers' states. SCI operation resumes after an interrupt brings the CPU out of stop mode. Exiting stop mode by reset aborts any transmission or reception in progress and resets the SCI.
- If the SCI stop disable bit is set, the SCI operates normally in stop mode.

33.5 Resets

Any system reset completely resets the SCI.

33.6 Clocks

All timing is derived from the IP Bus clock, which is the main clock for this module. See Baud-Rate Generation about how the data rate is determined.

33.7 Interrupts

Table 33-10. Interrupt Summary

Interrupt	Source	Description
TDRE	Transmitter	Transmit Data Register Empty interrupt
TIDLE	Transmitter	Transmit Idle interrupt
RERR	Receiver	Receive Error (FE, NF, PF, LSE, or OR) interrupt
RDRF/OR	Receiver	Receive Data Register Full / Overrun / Active Edge interrupt

33.7.1 Description of Interrupt Operation

Table 33-11. SCI Interrupt Sources

Interrupt Source	Flag	Local Enable	Description
Transmitter	STAT[TDRE]	CTRL1[TEIE]	Transmit Data Register Empty interrupt
Transmitter	STAT[TIDLE]	CTRL1[TIIE]	Transmit Idle interrupt
Receiver	STAT[RDRF] STAT[OR]	CTRL1[RFIE]	Receive Data Register Full / Overrun / Active Edge interrupt
	STAT[RIEF]	CTRL2[RIEIE]	
Receiver	STAT[FE] STAT[NF]	CTRL1[REIE]	Receive Error (FE, NF, PF, LSE, or OR) interrupt

Table 33-11. SCI Interrupt Sources

Interrupt Source	Flag	Local Enable	Description
	STAT[PF] STAT[LSE] STAT[OR]		

33.7.1.1 Transmitter Empty Interrupt

The transmitter empty interrupt is enabled by setting CTRL1[TEIE]. When this interrupt is enabled, an interrupt is generated while data is transferred from the SCI data register to the transmit shift register. The interrupt service routine should read the STAT register, verify that STAT[TDRE] is set, and write the next data to be transmitted to the DATA register, which clears STAT[TDRE].

NOTE

This interrupt is disabled if CTRL2[TDE] is set, enabling transmit DMA operations.

33.7.1.2 Transmitter Idle Interrupt

The transmitter idle interrupt is enabled by setting CTRL1[TIIE]. This interrupt indicates that STAT[TIDLE] is set and the transmitter is no longer sending data, preamble, or break characters. The interrupt service routine should read the STAT register, verify STAT[TIDLE] is set, and initiate a preamble, break, or write a data character to the DATA register. Any of these actions clears STAT[TIDLE] because the transmitter is then busy.

33.7.1.3 Receiver Full Interrupt

The receiver full interrupt is enabled by setting CTRL1[RFIE]. This interrupt indicates that receive data is available in the DATA register. The interrupt service routine should read the STAT register, verify that STAT[RDRF] is set, and then read the data from the DATA register, which clears STAT[RDRF].

NOTE

This interrupt is disabled if CTRL2[RDE] is set, enabling receive DMA operations.

33.7.1.4 Receiver Edge Interrupt

This interrupt is used signal that an active edge has been observed on the RXD input pin. An interrupt is generated only when enabled using the CTRL2[RIEIE] bit. The CTRL[POL] bit determines which logic state is considered active. This interrupt is typically used to wake the part from stop mode.

33.7.1.5 Receive Error Interrupt

The receive error interrupt is enabled by setting CTRL1[REIE]. This interrupt indicates that the receiver detected any of the errors reflected by the following conditions:

- Noise flag (STAT[NF]) set
- Parity error flag (STAT[PF]) set
- Framing error flag (STAT[FE]) set
- Overrun flag (STAT[OR]) set
- LIN sync error flag (STAT[LSE]) set

The interrupt service routine should read the STAT register to determine which of the error flags was set. The error flag is cleared by writing (anything) to the STAT register. Then software should take the appropriate action to handle the error condition.

33.7.2 Recovery from Wait and Stop Mode

Any enabled SCI interrupt request can bring the core out of wait mode or stop mode (if the SCI module is enabled in stop mode).

33.8 DMA Requests

Table 33-12. SCI DMA Request Sources

DMA Request Source	Flag	Local Enable	Description
Transmitter	STAT[TDMA]	CTRL2[TDE]	Transmit Data Write Request
Receiver	STAT[RDMA]	CTRL2[RDE]	Receive Data Read Request

33.8.1 Transmit Data Write Request

This request is enabled by setting CTRL2[TDE]. Once enabled, the request is generated when there is an empty space in the TX data register/FIFO. The DMA controller should write data to the register/FIFO until full.

33.8.2 Receive Data Read Request

This request is enabled by setting CTRL2[RDE]. Once enabled, the request is generated when there is data in the RX data register/FIFO. The DMA controller should read the data register/FIFO until empty.

Chapter 34
Queued Serial Peripheral Interface (QSPI)

34.1 Introduction

34.1.1 Overview

The serial peripheral interface (SPI) module enables full-duplex, synchronous, serial communication between the chip and peripheral devices, including other chips. Software can poll the SPI status flags or SPI operation can be interrupt driven. The block contains six 16-bit memory mapped registers for control parameters, status, and data transfer.

Features of the SPI module include the following:

- Full-duplex operation
- Master and slave modes
- Double-buffered operation with separate transmit and receive registers
- Programmable Length Transactions (2 to 16 bits)
- Programmable transmit and receive shift order (MSB or LSB first)
- Fourteen master mode frequencies (maximum = bus frequency ÷ 2)
- Maximum slave mode frequency = bus frequency ÷ 4
- Serial clock with programmable polarity and phase
- Two separately enabled interrupts:
    - SPRF (SPI receiver full)
    - SPTE (SPI transmitter empty)
- Mode fault error flag with interrupt capability
- Overflow error flag with interrupt capability

- Wired OR mode functionality enabling connection to multiple SPIs
- Stop mode holdoff
- Separate RX and TX FIFO capable of handling 4 transactions

Maximum SPI data rates are limited by I/O pad performance as specified in the device's data sheet.

34.1.2 Block Diagram

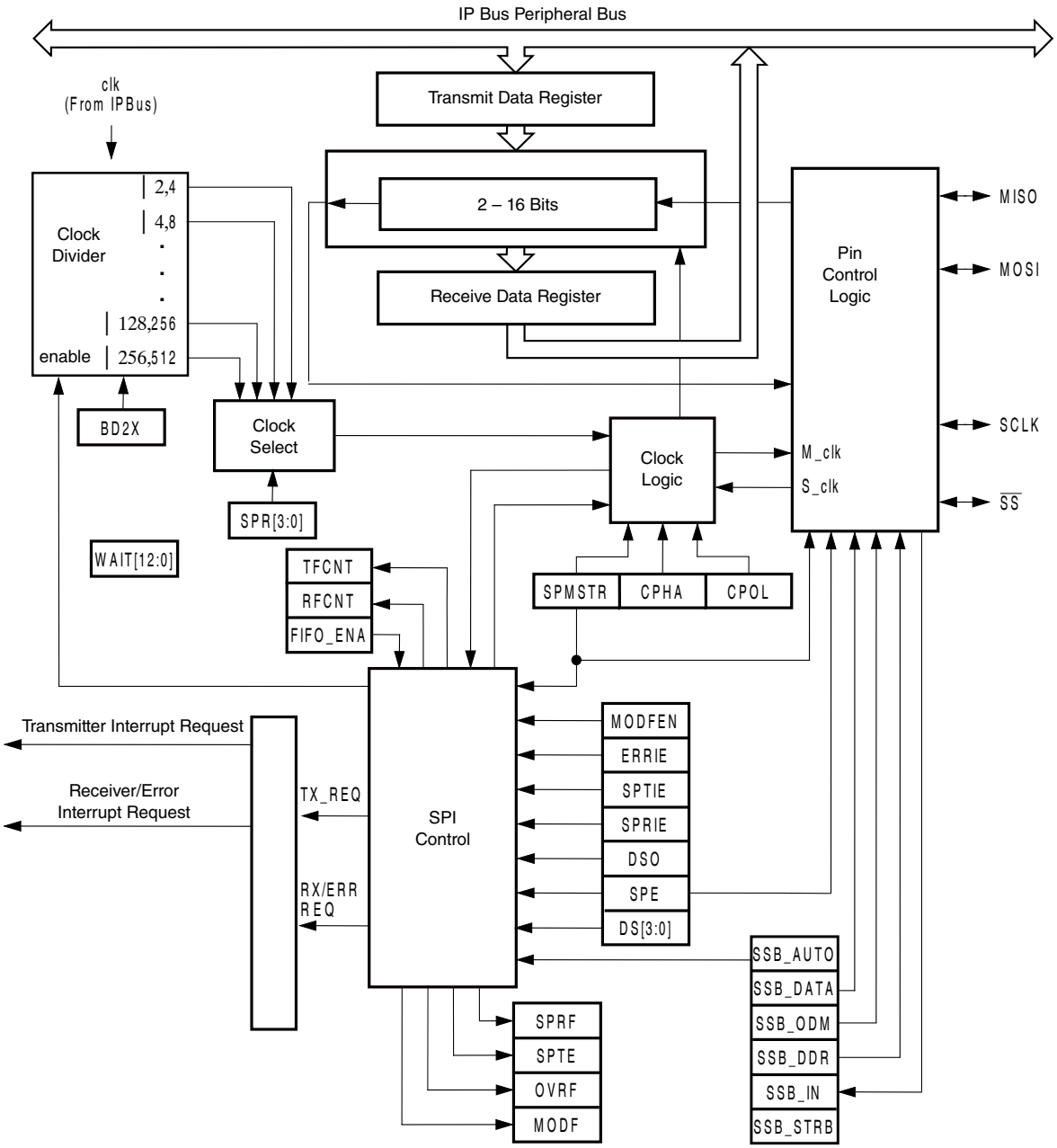


Figure 34-1. SPI Block Diagram

34.2 Signal Descriptions

34.2.1 External I/O Signals

The following are external I/O signals at the device interface. All of these pins are bidirectional.

Table 34-1. External I/O

Signal Name	Description	Direction	
		Master	Slave
MOSI	Master-out Slave-in Pad Pin	Output	Input
MISO	Master-in Slave-out Pad Pin	Input	Output
SCLK	Slave Clock Pad Pin	Output	Input
$\overline{SS}$	Slave Select Pad Pin (Active Low)	See Table 34-2.	

34.2.1.1 MISO (Master In/Slave Out)

MISO is one of the two SPI module pins that transmit serial data. In full duplex operation, the MISO pin of the master SPI module is connected to the MISO pin of the slave SPI module. The master SPI simultaneously receives data on its MISO pin and transmits data from its MOSI pin.

Slave output data on the MISO pin is enabled only when the SPI is configured as a slave. The SPI is configured as a slave when its SPMSTR bit (see the description of the SPI status and control register) is logic zero and its SS pin is at logic zero. To support a multiple-slave system, a logic one on the $\overline{SS}$ pin puts the MISO pin in a high-impedance state.

34.2.1.2 MOSI (Master Out/Slave In)

MOSI is one of the two SPI module pins that transmits serial data. In full-duplex operation, the MOSI pin of the master SPI module is connected to the MOSI pin of the slave SPI module. The master SPI simultaneously transmits data from its MOSI pin and receives data on its MISO pin.

34.2.1.3 SCLK (Serial Clock)

The serial clock synchronizes data transactions between master and slave devices. In a master device, the SCLK pin is the clock output. In a slave device, the SCLK pin is the clock input. In full duplex operation, the master and slave devices exchange data in the same number of clock cycles as the number of bits of transmitted data.

34.2.1.4 $\overline{SS}$ (Slave Select)

The $\overline{SS}$ pin has various functions depending on the current state of the SPI. For an SPI configured as a slave, the $\overline{SS}$ is used to select a slave. For CPHA = 0, the $\overline{SS}$ is used to define the start of a transaction. Because it is used to indicate the start of a transaction, the $\overline{SS}$ must be toggled high and low between each full length set of data transmitted for the CPHA = 0 format. However, it can remain low between transactions for the CPHA = 1 format.

When an SPI is configured as a slave, the $\overline{SS}$ pin is always configured as an input. The MODFEN bit can prevent the state of the $\overline{SS}$ from creating a MODF error.

NOTE

A logic one voltage on the $\overline{SS}$ pin of a slave SPI puts the MISO pin in a high-impedance state. The slave SPI ignores all incoming SCLK clocks, even if it is already in the middle of a transaction. A mode fault occurs if the $\overline{SS}$ pin changes state during a transaction.

When an SPI is configured as a master, the $\overline{SS}$ input can be used in conjunction with the MODF flag to prevent multiple masters from driving MOSI and SCLK. For the state of the $\overline{SS}$ pin to set the MODF flag, the MODFEN bit in the SPI Status and Control register must be set.

Table 34-2. SPI IO Configuration

SPE	SPMSTR	MODFEN	SPI CONFIGURATION	STATE OF SS_B LOGIC
0	X[1]	X	Not Enabled	$\overline{SS}$ ignored by SPI
1	0	X	Slave	Input-only to SPI
1	1	0	Master without MODF	$\overline{SS}$ input ignored by SPI, $\overline{SS}$ output may be activated under software or hardware control to select slave devices.
1	1	1	Master with MODF	Input-only to SPI

1. X = don't care

34.3 Memory Map Registers

Six registers control and monitor QSPI module operation. These registers should be accessed only with word accesses. Accesses with lengths other than word lengths result in undefined results. Before QSPI registers can be changed, the bit corresponding to the QSPI must be set to 1 in the appropriate PCE register of the SIM.

QSPI memory map

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
E0B0	SPI Status and Control Register (QSPI_SPSCR)	16	R/W	6141h	34.3.1/904
E0B1	SPI Data Size and Control Register (QSPI_SPDSR)	16	R/W	See section	34.3.2/907
E0B2	SPI Data Receive Register (QSPI_SPDRR)	16	R	0000h	34.3.3/910
E0B3	SPI Data Transmit Register (QSPI_SPDTR)	16	W	0000h	34.3.4/911
E0B4	SPI FIFO Control Register (QSPI_SPFIFO)	16	R/W	000Ch	34.3.5/913
E0B5	SPI Word Delay Register (QSPI_SPWAIT)	16	R/W	0000h	34.3.6/915
E0B6	SPI Control Register 2 (QSPI_SPCTL2)	16	R/W	0000h	34.3.7/915

34.3.1 SPI Status and Control Register (QSPI_SPSCR)

This register does the following:

- Selects master SPI baud rate
- Determines data shift order
- Enables SPI module interrupt requests
- Configures the SPI module as master or slave
- Selects serial clock polarity and phase

NOTE

Using BFCLR or BFSET instructions on this register can cause unintended side effects on the status bits.

Address: E0B0h base + 0h offset = E0B0h

Bit	15	14	13	12	11	10	9	8
Read Write	SPR[2:0]			DSO	ERRIE	MODFEN	SPRIE	SPMSTR
Reset	0	1	1	0	0	0	0	1

Bit	7	6	5	4	3	2	1	0
Read	CPOL	CPHA	SPE	SPTIE	SPRF	OVRF	MODF	SPTE
Write								
Reset	0	1	0	0	0	0	0	1

QSPI_SPSCR field descriptions

Field	Description

15–13
SPR[2:0]
	SPI Baud Rate Select
	
	In master mode, these read/write bits select one of eight baud rates. SPR2, SPR1, and SPR0 have no effect in slave mode. Reset sets SPR[2:0] to b011.
	
	Use the following formula to calculate the SPI baud rate:
	
	`Baud rate = clk/BD`
	
	where:
	
	`clk = Peripheral Bus Clock`
	`BD = baud rate divisor`
	
	Restriction: The maximum data transmission rate for the SPI is typically limited by the bandwidth of the I/O drivers on the chip, which can be a function of manufacturing technology. The typical limit in Normal mode is 40 MHz and in Wired-OR mode is 10 MHz. These baud rate limitations apply to both master and slave mode. The value of BD must be set to ensure the module remains within these ranges.
	
	NOTE: The value of BD can also depend on the values of the SPR3 and BD2X fields in the SPI Data Size and Control Register.
	
	000 BD = 2 when SPR3 = 0, BD = 512 when SPR3 = 1 (double BD when BD2X = 1)
	001 BD = 4 when SPR3 = 0, BD = 1024 when SPR3 = 1 (double BD when BD2X = 1)
	010 BD = 8 when SPR3 = 0, BD = 2048 when SPR3 = 1 (double BD when BD2X = 1)
	011 BD = 16 when SPR3 = 0, BD = 4096 when SPR3 = 1 (double BD when BD2X = 1)
	100 BD = 32 when SPR3 = 0, BD = 8192 when SPR3 = 1 (double BD when BD2X = 1)
	101 BD = 64 when SPR3 = 0 (double BD when BD2X = 1), BD = 16384 when SPR3 = 1 (regardless of BD2X)
	110 BD = 128 when SPR3 = 0 (double BD when BD2X = 1), BD = 16384 when SPR3 = 1 (regardless of BD2X)
	111 BD = 256 when SPR3 = 0 (double BD when BD2X = 1), BD = 16384 when SPR3 = 1 (regardless of BD2X)

12 DSO	Data Shift Order This read/write bit determines which bit is transmitted or received first, either the MSB or LSB. Both master and slave SPI modules must transmit and receive packets of the same length. Regardless of how this bit is set, when reading from the data receive register or writing to the data transmit register, the LSB is always at bit location 0 and the MSB is at the correct bit position. If the data length is less than 16 bits, the upper bits of data are zero padded. 0 MSB transmitted first (MSB -> LSB) 1 LSB transmitted first (LSB -> MSB)
11 ERRIE	Error Interrupt Enable This read/write bit enables the MODF (if MODFEN is also set) and OVRF bits to generate device interrupt requests. Reset clears the ERRIE bit.

Table continues on the next page...

QSPI_SPSCR field descriptions (continued)

Field	Description
	0 MODF and OVRF cannot generate device interrupt requests 1 MODF and OVRF can generate device interrupt requests
10 MODFEN	Mode Fault Enable This read/write bit, when set to 1, allows the MODF flag to be set. If the MODF flag is set, clearing the MODFEN bit does not clear the MODF flag. If the MODFEN bit is low, the level of the SS_B pin does not affect the operation of an enabled SPI configured as a master. If configured as a master and MODFEN=1, a transaction in progress will stop if SS_B goes low. For an enabled SPI configured as a slave, having MODFEN low only prevents the MODF flag from being set. It does not affect any other part of SPI operation.
9 SPRIE	SPI Receiver Interrupt Enable This read/write bit enables interrupt requests generated by the SPRF bit or the receive FIFO watermark register. 0 SPRF interrupt requests disabled 1 SPRF interrupt requests enabled
8 SPMSTR	SPI Master This read/write bit selects master mode operation or slave mode operation. 0 Slave mode 1 Master mode
7 CPOL	Clock Polarity This read/write bit determines the logic state of the SCLK pin between transactions. To transmit data between SPI modules, the SPI modules must have identical CPOL values. 0 Rising edge of SCLK starts transaction 1 Falling edge of SCLK starts transaction
6 CPHA	Clock Phase This read/write bit controls the timing relationship between the serial clock and SPI data. To transmit data between SPI modules, the SPI modules must have identical CPHA values. When CPHA = 0, the SS_B pin of the slave SPI module must be set to 1 between data words. To set SSB to 1 between data words when SSB_AUTO is 1, set SSB_STRB to 1. **Restriction:** *Do not* use CPHA = 0 while in DMA mode.
5 SPE	SPI Enable This read/write bit enables the SPI module. Clearing SPE causes a partial reset of the SPI. **Restriction:** When you change the SPE bit, the write statement must change *only* the SPE bit. Change any other bits in a separate write statement. In master mode the SPE bit can be cleared by a mode fault condition. 0 SPI module disabled 1 SPI module enabled
4 SPTIE	Transmit Interrupt Enable This read/write bit enables interrupt requests generated by the SPTE bit or the transmit FIFO watermark register.

Table continues on the next page...

QSPI_SPSCR field descriptions (continued)

Field	Description
	0 SPTE interrupt requests disabled 1 SPTE interrupt requests enabled
3 SPRF	SPI Receiver Full This clearable, read-only flag is set each time data transfers from the shift register to the data receive register and no space is available in the RX queue to receive new data (RX FIFO is full). SPRF generates an interrupt request if the SPRIE bit in the SPI control register is set. This bit automatically clears after the data receive register is read. 0 Receive data register or FIFO is not full. (If using the FIFO, read RFCNT to determine the number of valid words available.) 1 Receive data register or FIFO is full.
2 OVRF	Overflow This clearable, read-only flag is set if software does not read the data in the receive data register before the next full data enters the shift register. In an overflow condition, the data already in the receive data register is unaffected, and the data shifted in last is lost. Clear the OVRF bit by reading the SPI status and control register with OVRF set and then reading the receive data register. 0 No overflow 1 Overflow
1 MODF	Mode Fault This clearable, read-only flag is set in a slave SPI if the SS_B pin goes high during a transaction with the MODFEN bit set. In a master SPI, the MODF flag is set if the SS_B pin goes low at any time with the MODFEN bit set. Clear the MODF bit by writing a one to the MODF bit when it is set. 0 SS_B pin at appropriate logic level 1 SS_B pin at inappropriate logic level
0 SPTE	SPI Transmitter Empty This clearable, read-only flag is set each time the transmit data register transfers data into the shift register and there is no more new data available in the TX queue (TX FIFO is empty). SPTE generates an interrupt request if the SPTIE bit in the SPI control register is set. SPTE is cleared by writing to the data transmit register. **CAUTION:** Do not write to the SPI data register unless the SPTE bit is high. Otherwise, data may be lost. 0 Transmit data register or FIFO is not empty. (If using the FIFO, read TFCNT to determine how many words can be written safely.) 1 Transmit data register or FIFO is empty.

34.3.2 SPI Data Size and Control Register (QSPI_SPDSR)

This read/write register determines the data length for each transaction. The master and slave must transfer the same size data on each transaction. A new value takes effect only at the time the SPI is enabled (the SPE bit in the status and control register is set from 0 to 1). To have a new value take effect, disable and then re-enable the SPI with the new value in the register.

To use the SS_B control functions in master mode, set the appropriate bit in a GPIO peripheral enable register to enable peripheral control of the SS_B pin.

Address: E0B0h base + 1h offset = E0B1h

Bit	15	14	13	12	11	10	9	8
Read	WOM	TDMAEN	RDMAEN	BD2X	SSB_IN	SSB_DATA	SSB_ODM	SSB_AUTO
Write								
Reset	0	0	0	0	x*	1	0	0

Bit	7	6	5	4	3	2	1	0
Read	SSB_DDR	SSB_STRB	SSB_OVER	SPR3	DS[3:0]			
Write								
Reset	0	0	0	0	1	1	1	1

* Notes:
- x = Undefined at reset.

QSPI_SPDSR field descriptions

Field	Description
15 WOM	Wired-OR Mode The Wired-OR mode (WOM) control bit is used to select the nature of the SPI pins. When enabled (the WOM bit is set), the SPI pins are configured as open-drain drivers. When disabled (the WOM bit is cleared), the SPI pins are configured as push-pull drivers. 0 The SPI pins are configured as push-pull drivers. 1 The SPI pins are configured as open-drain drivers with the pull-ups disabled.
14 TDMAEN	Transmit DMA Enable This read/write bit enables DMA control for transmit data.
13 RDMAEN	Receive DMA Enable This read/write bit enables DMA control for receive data.
12 BD2X	Baud Divisor Times Setting this bit causes the Baud Rate Divisor (BD) to be multiplied by two. The SPR bits define BD.
11 SSB_IN	SS_B Input This read only bit shows the current state of the SS_B pin in all modes. The bit's reset state is undefined.
10 SSB_DATA	SS_B Data This read/write bit is the value to drive on the SS_B pin. This bit is disabled when SSB_AUTO=1 or SSB_STRB=1. 0 SS_B pin is driven low if SSB_DDR=1 1 SS_B pin is driven high if SSB_DDR=1
9 SSB_ODM	SS_B Open Drain Mode This read/write bit enables open drain mode on the SS_B pin in master mode.

Table continues on the next page...

QSPI_SPDSR field descriptions (continued)

Field	Description
	0 SS_B is configured for high and low drive. This mode is generally used in single master systems. 1 SS_B is configured as an open drain pin (only drives low output level). This mode is useful for multiple master systems.
8 SSB_AUTO	SS_B Automatic Mode This read/write bit enables hardware control of the SS_B pin in master mode. (The legacy design requires software to control the SS_B output pin.) The initial falling edge of SS_B is generated and SS_B is held low until the TX buffer or FIFO is empty. This bit may be used alone or in combination with SS_STRB to generate the required SS_B signal. 0 SS_B output signal is software generated by directly manipulating the various bits in this register or the GPIO registers (compatible with legacy SPI software). 1 SS_B output signal is hardware generated to create the initial falling edge and final rising edge. The idle state of the SS_B is high. **Restriction:** Do not use if MODFEN = 1.
7 SSB_DDR	SS_B Data Direction This read/write bit controls input/output mode on the SS_B pin in master mode. 0 SS_B is configured as an input pin. Use this setting in slave mode or in master mode with MODFEN=1. 1 SS_B is configured as an output pin. Use this setting in master mode with MODFEN=0.
6 SSB_STRB	SS_B Strobe Mode This read/write bit enables hardware pulse of the SS_B pin in master mode between words. This bit may be used alone or in combination with the SS_AUTO to generate the required SS_B signal. Pulses are generated between words irrespective of the setting of CPHA. 0 No SS_B pulse between words. 1 SS_B output signal is pulsed high between words. This adds 1.5 baud clocks to the total word period. The idle state of SS_B is low unless SSB_AUTO is high and then the idle state is high. **Restriction:** Do not use if MODFEN = 1.
5 SSB_OVER	SS_B Override This read/write bit overrides the internal SS_B signal input from the I/O pad and replaces it with a level equal to the setting of the SPMSTR bit. This allows the SPI to function in slave mode, when CPHA=1, without commiting a GPIO pin to be tied low. **Restriction:** This bit should not be used in multi-slave systems or when CPHA=0. **Restriction:** This bit should not be used in a multi-master system because in master mode a mode fault error cannot be generated. 0 SS_B internal module input is selected to be connected to a GPIO pin. 1 SS_B internal module input is selected to be equal to SPMSTR.
4 SPR3	SPI Baud Rate Select Use this bit with SPR[2:0] in the status and control register and BD2X to define the Baud Rate Divisor (BD).
DS[3:0]	Transaction data size 4'h0 Not allowed

Table continues on the next page...

QSPI_SPDSR field descriptions (continued)

Field	Description
	4'h1 2 bits transaction data size 4'h2 3 bits transaction data size 4'h3 4 bits transaction data size 4'h4 5 bits transaction data size 4'h5 6 bits transaction data size 4'h6 7 bits transaction data size 4'h7 8 bits transaction data size 4'h8 9 bits transaction data size 4'h9 10 bits transaction data size 4'hA 11 bits transaction data size 4'hB 12 bits transaction data size 4'hC 13 bits transaction data size 4'hD 14 bits transaction data size 4'hE 15 bits transaction data size 4'hF 16 bits transaction data size

34.3.3 SPI Data Receive Register (QSPI_SPDRR)

The SPI data receive register is read-only. Reading data from the register shows the last data received after a complete transaction. The SPRF bit is set when new data is transferred to this register.

Address: E0B0h base + 2h offset = E0B2h

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	R15	R14	R13	R12	R11	R10	R9	R8	R7	R6	R5	R4	R3	R2	R1	R0
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

QSPI_SPDRR field descriptions

Field	Description
15 R15	Receive Data Bit 15
14 R14	Receive Data Bit 14
13 R13	Receive Data Bit 13
12 R12	Receive Data Bit 12
11 R11	Receive Data Bit 11

Table continues on the next page...

QSPI_SPDRR field descriptions (continued)

Field	Description
10 R10	Receive Data Bit 10
9 R9	Receive Data Bit 9
8 R8	Receive Data Bit 8
7 R7	Receive Data Bit 7
6 R6	Receive Data Bit 6
5 R5	Receive Data Bit 5
4 R4	Receive Data Bit 4
3 R3	Receive Data Bit 3
2 R2	Receive Data Bit 2
1 R1	Receive Data Bit 1
0 R0	Receive Data Bit 0

34.3.4 SPI Data Transmit Register (QSPI_SPDTR)

The SPI data transmit register is write-only. Writing data to this register writes the data to the transmit data buffer. When the SPTE bit is set, new data should be written to this register. If new data is not written while in master mode, a new transaction will not begin until this register is written.

When selected in slave mode, the old data will be re-transmitted. When *not* selected and in slave mode, transmit data will remain unchanged. All data should be written with the LSB at bit 0. This register can only be written when the SPI is enabled (SPE = 1).

Address: E0B0h base + 3h offset = E0B3h

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read																
Write	T15	T14	T13	T12	T11	T10	T9	T8	T7	T6	T5	T4	T3	T2	T1	T0
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

QSPI_SPDTR field descriptions

Field	Description
15 T15	Transmit Data Bit 15
14 T14	Transmit Data Bit 14
13 T13	Transmit Data Bit 13
12 T12	Transmit Data Bit 12
11 T11	Transmit Data Bit 11
10 T10	Transmit Data Bit 10
9 T9	Transmit Data Bit 9
8 T8	Transmit Data Bit 8
7 T7	Transmit Data Bit 7
6 T6	Transmit Data Bit 6
5 T5	Transmit Data Bit 5
4 T4	Transmit Data Bit 4
3 T3	Transmit Data Bit 3
2 T2	Transmit Data Bit 2
1 T1	Transmit Data Bit 1
0 T0	Transmit Data Bit 0

34.3.5 SPI FIFO Control Register (QSPI_SPFIFO)

This register is used for FIFO control and status.

Address: E0B0h base + 4h offset = E0B4h

Bit	15	14	13	12	11	10	9	8
Read	0	TFCNT			0	RFCNT		
Write								
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
Read	0	TFWM		0	RFWM		0	FIFO_ENA
Write								
Reset	0	0	0	0	1	1	0	0

QSPI_SPFIFO field descriptions

Field	Description
15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
14–12 TFCNT	TX FIFO Level These read-only bits show how many words are used in the TX FIFO. Writes to the data transmit register cause TFCNT to increment, and, as words are pulled for transmission, TFCNT is decremented. Attempts to write new data to the data transmit register are ignored when TFCNT indicates the FIFO is full. If master mode is enabled, transmission continues until the FIFO is empty, even if SPE is set to 0. 000 Tx FIFO empty (if enabled Transmit Empty Interrupt asserted) 001 One word used in Tx FIFO 010 Two words used in Tx FIFO 011 Three words used in Tx FIFO 100 Tx FIFO full
11 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
10–8 RFCNT	RX FIFO Level These read-only bits show how many words are used in the RX FIFO. As words are received, the value of RFCNT is incremented; as words are read from the data receive register, the value of RFCNT is decremented. There is one word time to read the data receive register between when the SPRF status bit is set (interrupt asserted) and when an overflow condition is flagged. 000 Rx FIFO empty 001 One word used in Rx FIFO 010 Two words used in Rx FIFO 011 Three words used in Rx FIFO 100 Rx FIFO full (if enabled Receiver Full Interrupt asserted)

Table continues on the next page...

QSPI_SPFIFO field descriptions (continued)

Field	Description
7 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
6–5 TFWM	Tx FIFO Watermark These read/write bits determine how many words must remain in the Tx FIFO before an interrupt is generated. Increasing the value of TFWM increases the allowable latency in servicing the Tx interrupt without underrunning the Tx buffer space. Larger values of TFWM may also increase the number of Tx interrupt service requests because the maximum number of Tx words may not be available when the service routine is activated. If TFWM is set to the minimum value then only one SPI word time in interrupt service latency is allowed before an underrun condition results and continuous transmission is stopped in master mode or the last data word is re-transmitted in slave mode. This field is ignored when FIFO_ENA = 0. To clear an interrupt generated by TFWM, new words must be written to the data transmit register or the value of TFWM must be reduced. 00 Transmit interrupt active when Tx FIFO is empty 01 Transmit interrupt active when Tx FIFO has one or fewer words available 10 Transmit interrupt active when Tx FIFO has two or fewer words available 11 Transmit interrupt active when Tx FIFO has three or fewer words available
4 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
3–2 RFWM	Rx FIFO Watermark These read/write bits determine how many words must be used in the Rx FIFO before an interrupt is generated. Decreasing the value of RFWM increases the allowable latency in servicing the Rx interrupt without overrunning the Rx buffer space. Smaller values of RFWM may also increase the number of Rx interrupt service requests because the maximum number of Rx words may not have been used when the service routine is activated. If RFWM is set to the maximum value then only one SPI word time in interrupt service latency is allowed before an overrun condition results and receive data is lost. This field is ignored when FIFO_ENA = 0. To clear an interrupt generated by RFWM, words must be read from the data receive register or the value of RFWM must be increased. 00 Receive interrupt active when Rx FIFO has at least one word used 01 Receive interrupt active when Rx FIFO has at least two words used 10 Receive interrupt active when Rx FIFO has at least three words used 11 Receive interrupt active when Rx FIFO is full
1 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
0 FIFO_ENA	FIFO Enable This read/write bit enables Tx and Rx FIFOs' mode. 0 FIFOs are disabled and reset. 1 FIFOs are enabled. FIFOs retain their status even if SPE is set to 0.

34.3.6 SPI Word Delay Register (QSPI_SPWAIT)

This register is used to control the delay between words.

Address: E0B0h base + 5h offset = E0B5h

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	0			WAIT												
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

QSPI_SPWAIT field descriptions

Field	Description
15–13 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
WAIT	Wait Delay This 13-bit register controls the time between data transactions in master mode. It sets the delay between words to be a number of Peripheral Bus Clocks equal to (WAIT + 1). This delay is used only when a word is waiting to be transmitted at the completion of the transmission of the current word. If no word is waiting to be transmitted, the SPI goes idle at the completion of the current transmission and subsequently starts transmission immediately when a new word is written to the Data Transmit register.

34.3.7 SPI Control Register 2 (QSPI_SPCTL2)

This register controls the stop mode holdoff feature.

Address: E0B0h base + 6h offset = E0B6h

Bit	15	14	13	12	11	10	9	8
Read	0							
Write								
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
Read	0							SHEN
Write								
Reset	0	0	0	0	0	0	0	0

QSPI_SPCTL2 field descriptions

Field	Description
15–1 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

Table continues on the next page...

QSPI_SPCTL2 field descriptions (continued)

Field	Description
0 SHEN	Stop Mode Holdoff Enable When enabled, this bit allows the SPI module to hold off entry to chip level stop mode if a word is being transmitted or received. Stop mode will be entered after the SPI finishes transmitting/receiving. This bit does not allow the SPI to wake the chip from stop mode in any way. The SHEN bit can only delay the entry into stop mode. This bit should not be set in slave mode because the state of SS_B (which would be controlled by an external master device) may cause the logic to hold off stop mode entry forever. 0 Disable stop mode holdoff . 1 Enable stop mode holdoff while the SPI is transmitting/receiving.

34.4 Functional Description

34.4.1 Operating Modes

34.4.1.1 Master Mode

The SPI operates in master mode when the SPI master bit, SPMSTR, is set.

Note

Configure the SPI module as master or slave before enabling the SPI. Enable the master SPI before enabling the slave SPI. Disable the slave SPI before disabling the master SPI.

Only a master SPI module can initiate transactions. With the SPI enabled, software begins the transaction from the master SPI module by writing to the transmit data register. If the shift register is empty, the data immediately transfers to the shift register, setting the SPI transmitter empty bit, SPTE. The data begins shifting out on the MOSI pin under the control of the SPI serial clock, SCLK.

The SPR3, SPR2, SPR1, and SPR0 bits in the SPI registers control the baud rate generator and determine the speed of the shift register. Through the SCLK pin, the baud rate generator of the master also controls the shift register of the slave peripheral

As the data shifts out on the MOSI pin of the master, external data shifts in from the slave on the master's MISO pin. The transaction ends when the receiver full bit, SPRF, becomes set. At the same time that SPRF becomes set, the data from the slave transfers to the SPI Data Receive register. In normal operation, SPRF signals the end of a transaction. Software clears SPRF by reading the SPI Data Receive register. Writing to the SPI Data Transmit register clears the SPTE bit.

The following figure is an example configuration for a full-duplex master-slave configuration. Having the $\overline{SS}$ bit of the master device held high is only necessary if MODFEN = 1. Tying the slave $\overline{SS}$ bit to ground should only be done if CPHA = 1.

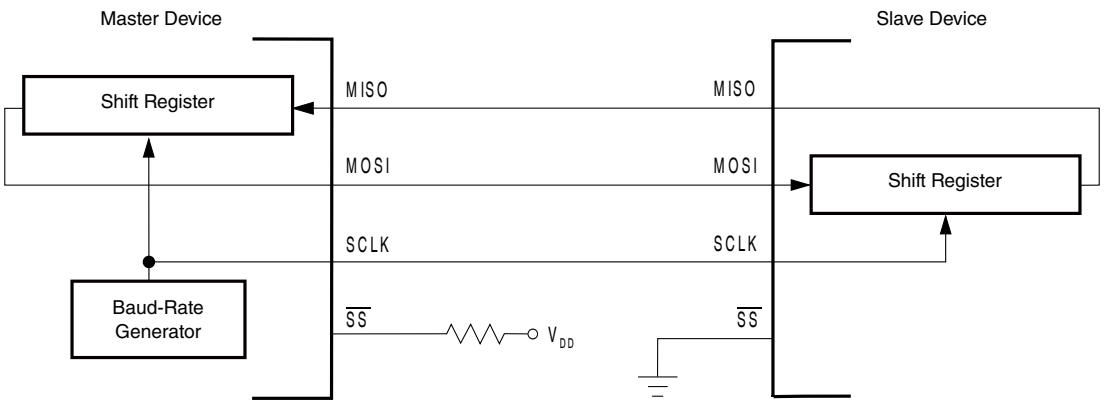


Figure 34-2. Full-Duplex Master-Slave Connections

34.4.1.2 Slave Mode

The SPI operates in slave mode when the SPMSTR bit is 0. In slave mode the SCLK pin is the input for the serial clock from the master device. Before a data transaction occurs, the SS pin of the slave SPI must be at logic zero. $\overline{SS}$ must remain low until the transaction completes or a mode fault error occurs.

Note

The SPI must be enabled (SPE = 1) for slave transactions to be received.

Note

Data in the transmitter shift register is unaffected by SCLK transitions when the SPI operates as a slave but is deselected ($\overline{SS}$ = 1).

In a slave SPI module, data enters the shift register under the control of the serial clock, SCLK, from the master SPI module. After a full data word enters the shift register of a slave SPI, it transfers to the SPI Data Receive register, and the SPRF bit is set. To prevent an overflow condition, slave software then must read the receive data register before another full data word enters the shift register.

The maximum frequency of the SCLK for an SPI configured as a slave is less than 1/2 the bus clock frequency. The frequency of the SCLK for an SPI configured as a slave does not have to correspond to any SPI baud rate as defined by the SPR bits. The SPR bits control only the speed of the SCLK generated by an SPI configured as a master.

When the master SPI starts a transaction, the data in the slave shift register begins shifting out on the MISO pin. The slave can load its shift register with new data for the next transaction by writing to its transmit data register. The slave must write to its transmit data register at least one bus cycle before the master starts the next transaction. Otherwise, the data that was last transmitted is reloaded into the slave shift register and shifts out on the MISO pin again. Data written to the slave shift register during a transaction remains in a buffer until the end of the transaction.

When the clock phase bit (CPHA) is set, the first edge of SCLK starts a transaction. When CPHA is clear, the falling edge of $\overline{SS}$ starts a transaction.

Note

SCLK must be in the proper idle state before the slave is enabled to preserve the proper SCLK, MISO, MOSI timing relationships.

34.4.1.3 DMA Mode

When the TDMAEN or RDMAEN bit is set to 1, the SPI operates in DMA mode. Normal SPTE and/or SPRF interrupts are suppressed. Instead, DMA requests are generated, signaling the DMA controller to read and write the SPDRR or SPDTR as needed. The Transmitter Write DMA request is set whenever there is an open location in the transmit FIFO. Similarly, the Receiver Read DMA request is set whenever the receive FIFO is not empty. The DMA requests are suppressed in the case of an OVRF or MODF interrupt, as shown in Figure 34-3. If the SPI is being used to send and receive data, both TDMAEN and RDMAEN should be enabled or disabled at the same time. If data is only being received or transmitted, TDMAEN or RDMAEN may remain inactive as appropriate. However, in this case, ERRIE must not be set to prevent the DMA request being masked by a mode-fault error or overflow error.

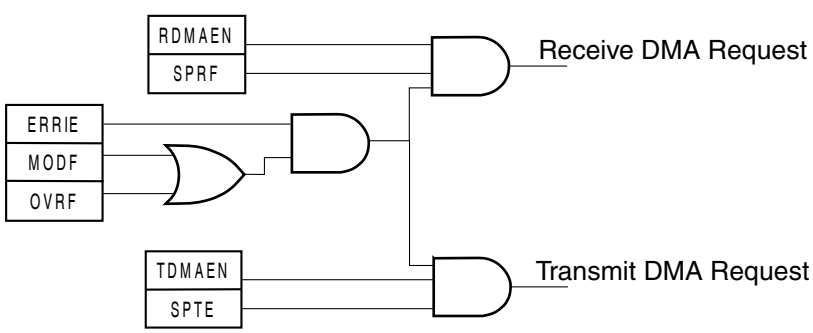


Figure 34-3. SPI DMA Request Generation

34.4.1.4 Wired-OR Mode

Wired-OR functionality is provided to permit the connection of multiple SPIs. The following figure illustrates the sharing of a single master device between multiple slave SPIs. When the WOM bit is set, the outputs switch from conventional complementary CMOS output to open drain outputs.

This internal pullup resistor brings the line high, and whichever SPI drives the line pulls it low as needed.

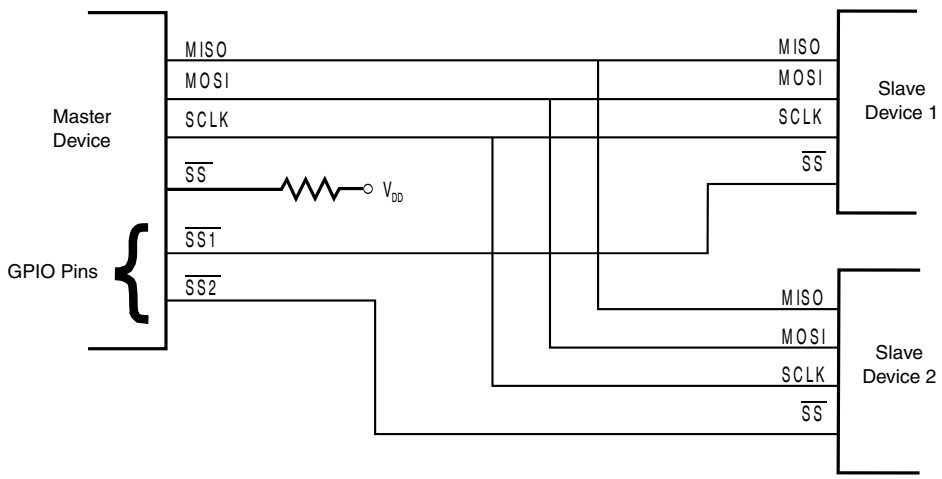


Figure 34-4. Master With Two Slaves

34.4.2 Transaction Formats

During an SPI transaction, data is simultaneously transmitted (shifted out serially) and received (shifted in serially). A serial clock synchronizes shifting and sampling on the two serial data lines. A slave select line enables selection of an individual slave SPI device. Slave devices that are not selected do not interfere with SPI bus activities. On a master SPI device, the slave select line can optionally be used to indicate multiple-master bus contention.

34.4.2.1 Data Transaction Length

The SPI can support data lengths of 2 to 16 bits. The length can be configured in the SPI Data Size and Control register. When the data length is less than 16 bits, the receive data register pads the upper bits with zeros.

Note

Data can be lost if the data length is not the same for both master and slave devices.

34.4.2.2 Data Shift Ordering

The SPI can be configured to transmit or receive the MSB of the desired data first or last, using the DSO bit in the SPI Status and Control register. Regardless of which bit is transmitted or received first, the data is always written to the SPI Data Transmit register and read from the SPI Data Receive register with the LSB in bit 0 and the MSB in correct position depending on the data transaction size.

34.4.2.3 Clock Phase and Polarity Controls

Software can select any of four combinations of serial clock (SCLK) phase and polarity using two bits in the SPI Status and Control register. The clock polarity is specified by the CPOL control bit, which selects an active high or low clock and has no significant effect on the transaction format.

The clock phase (CPHA) control bit selects one of two fundamentally different transaction formats. The 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 transactions, to enable a master device to communicate with peripheral slaves having different requirements.

Note

Before writing to the CPOL bit or the CPHA bit, disable the SPI (by clearing the SPI enable bit (SPE)).

34.4.2.4 Transaction Format When CPHA = 0

The following figure shows an SPI transaction in which CPHA is logic zero. The figure should not be used as a replacement for data sheet parametric information. It assumes 16 bit data lengths and the MSB shifted out first.

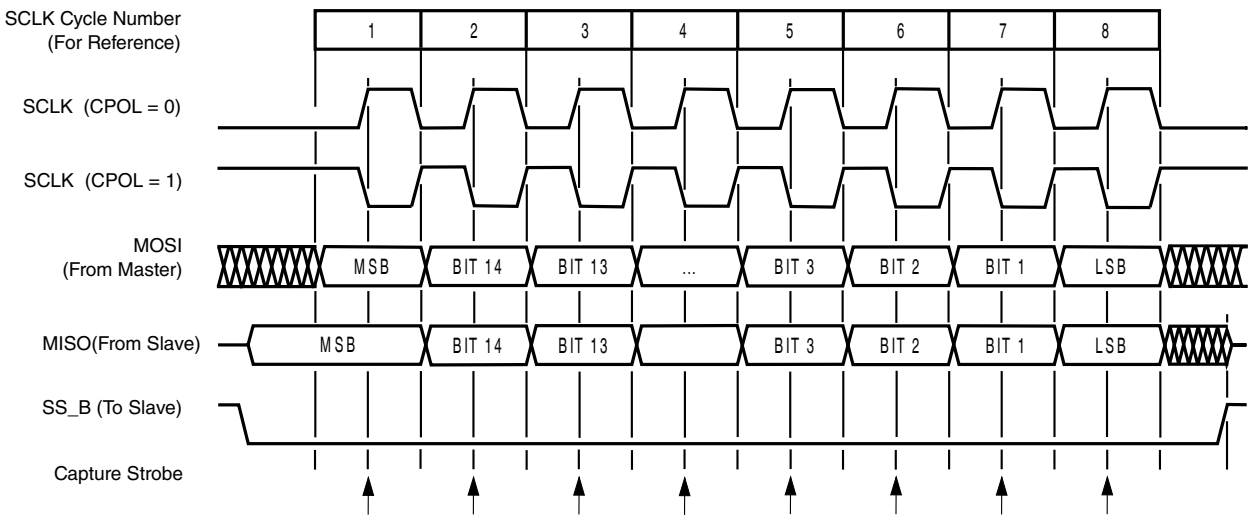


Figure 34-5. Transaction Format (CPHA = 0)

Two waveforms are shown for SCLK: one for CPOL = 0 and another for CPOL = 1. The diagram may be interpreted as a master or slave timing diagram since the serial clock (SCLK), master in/slave out (MISO), and master out/slave in (MOSI) pins are directly connected 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. When CPHA = 0, the first SCLK edge is the MSB capture strobe. Therefore, the slave must begin driving its data before the first SCLK edge, and a falling edge on the $\overline{SS}$ pin is used to start the slave data transaction. The slave $\overline{SS}$ pin must be toggled back to high and then low again between each data word transmitted, as shown in the following figure.

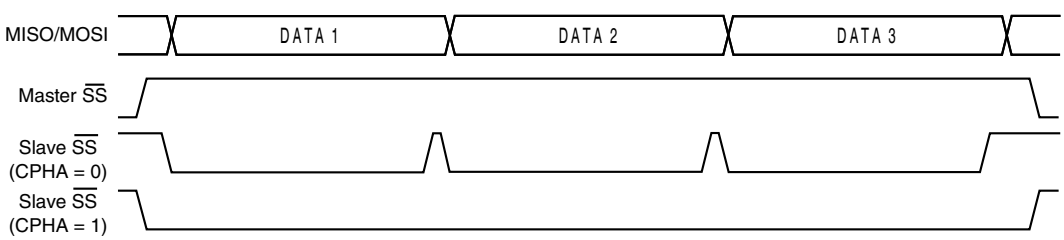


Figure 34-6. CPHA /$\overline{SS}$ Timing

When CPHA = 0 for a slave, the falling edge of $\overline{SS}$ indicates the beginning of the transaction. This causes the SPI to leave its idle state and begin driving the MISO pin with the first bit of its data. After the transaction begins, no new data is allowed into the shift register from the transmit data register. Therefore, the SPI data register of the slave must be loaded with transmit data before the falling edge of $\overline{SS}$. Any data written after the falling edge is stored in the transmit data register and transferred to the shift register after the current transaction. Also, for correct operation of the slave, SPE must be active before the negative edge of $\overline{SS}$ to correctly send/receive the first word. The $\overline{SS}$ line is the slave select input to the slave. The slave SPI drives its MISO output only when its slave select input ($\overline{SS}$) is at logic zero, so that only the selected slave drives to the master.

When CPHA = 0 for a master, normal operation would begin by the master initializing the $\overline{SS}$ pin of the slave high. A transfer would then begin by the master setting the $\overline{SS}$ pin of the slave low and then writing the SPI Data Transmit register. After a data transfer completes, the master device puts the $\overline{SS}$ pin back into the high state. While MODFEN = 1, the $\overline{SS}$ pin of the master must be high or a mode fault error occurs. If MODFEN = 0, the state of the $\overline{SS}$ pin is ignored.

34.4.2.5 Transaction Format When CPHA = 1

The following figure shows an SPI transaction in which CPHA is logic one. The figure should not be used as a replacement for data sheet parametric information. It assumes 16 bit data lengths and the MSB shifted out first.

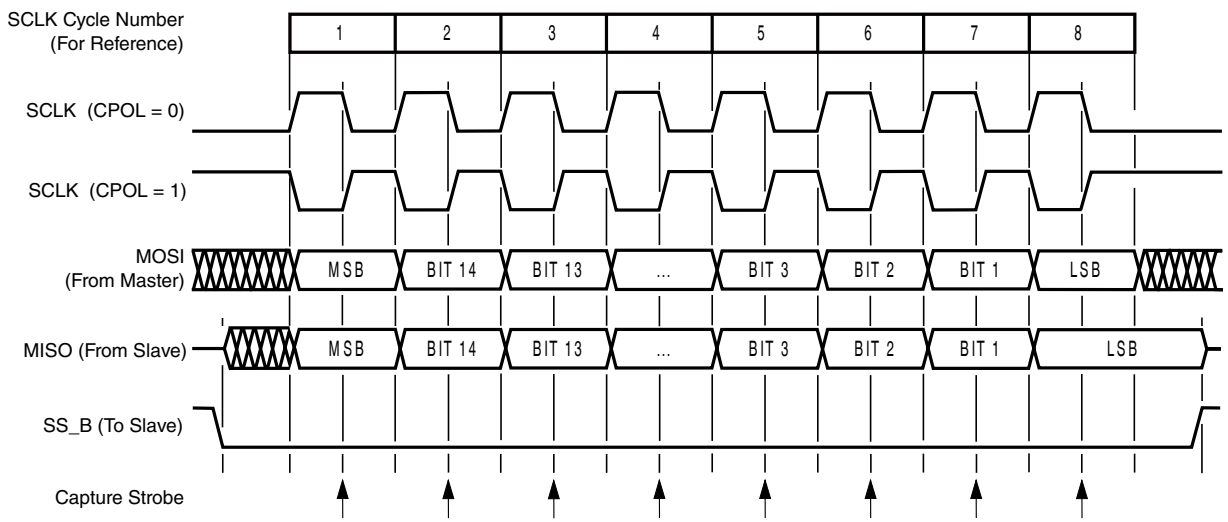


Figure 34-7. Transaction Format (CPHA = 1)

Two waveforms are shown for SCLK: one for CPOL = 0 and another for CPOL = 1. The diagram may be interpreted as a master or slave timing diagram since the serial clock (SCLK), master in/slave out (MISO), and master out/slave in (MOSI) pins are directly connected 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.

When CPHA = 1 for a slave, the first edge of the SCLK indicates the beginning of the transaction. This causes the SPI to leave its idle state and begin driving the MISO pin with the first bit of its data. After the transaction begins, no new data is allowed into the shift register from the transmit data register. Therefore, the SPI data register of the slave must be loaded with transmit data before the first edge of SCLK. Any data written after the first edge is stored in the transmit data register and transferred to the shift register after the current transaction. The $\overline{SS}$ line is the slave select input to the slave. The slave SPI drives its MISO output only when its slave select input ($\overline{SS}$) is at logic zero, so that only the selected slave drives to the master.

When CPHA = 1 for a master, the MOSI pin begins being driven with new data on the first SCLK edge. If MODFEN = 0 the $\overline{SS}$ pin of the master is ignored. Otherwise, the $\overline{SS}$ pin of the master must be high or a mode fault error occurs. The $\overline{SS}$ pin can remain low between transactions. This format may be preferable in systems with only one master and one slave driving the MISO data line.

34.4.2.6 Transaction Initiation Latency

When the SPI is configured as a master (SPMSTR is 1), writing to the SPI Data Transmit register starts a transaction. CPHA has no effect on the delay to the start of the transaction, but it does affect the initial state of the SCLK signal. When CPHA = 0, the SCLK signal remains inactive for the first half of the first SCLK cycle. When CPHA = 1, the first SCLK cycle begins with an edge on the SCLK line from its inactive to its active level. The SPI clock rate (selected by SPR2, SPR1, and SPR0) affects the delay from the write to the SPI Data Transmit register and the start of the SPI transaction. The internal baud clock in the master is a derivative of the internal device clock. To conserve power, it is enabled only after the SPMSTR bit is set and there is a new word written to the SPI Data Transmit register. If the SPI Data Transmit register has no new word when the current transaction completes, the internal baud clock is stopped. The initiation delay is a single SPI bit time, as the following figure shows. That is, the delay is 4 bus cycles for DIV4, 8 bus cycles for DIV8, 16 bus cycles for DIV16, 32 bus cycles for DIV32, and so on.

Note

The following figure assumes 16-bit data lengths and the MSB shifted out first.

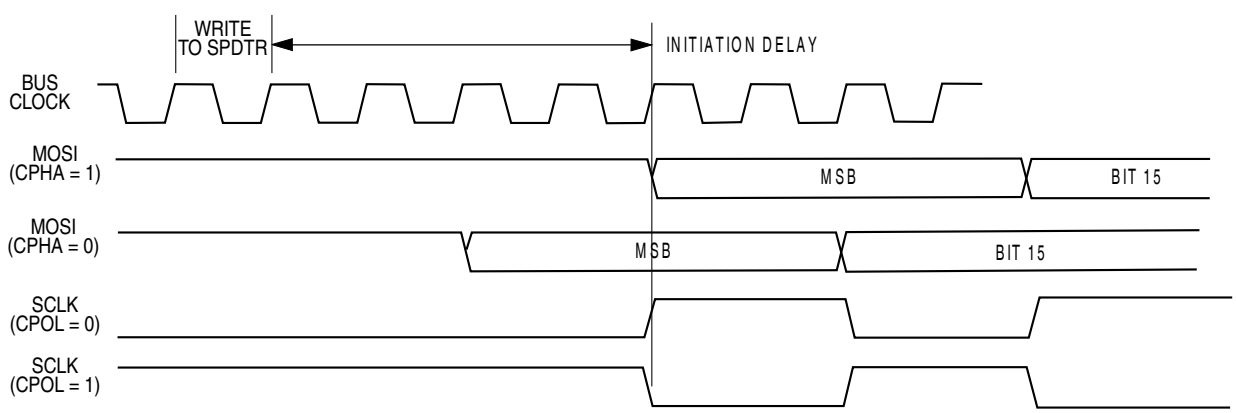


Figure 34-8. Transaction Start Delay (Master)

34.4.2.7 $\overline{SS}$ Hardware-Generated Timing in Master Mode

If the SSB_STRB bit is set in master mode, the SPI generates a word strobe pulse on $\overline{SS}$ for a slave device (see the following figures).

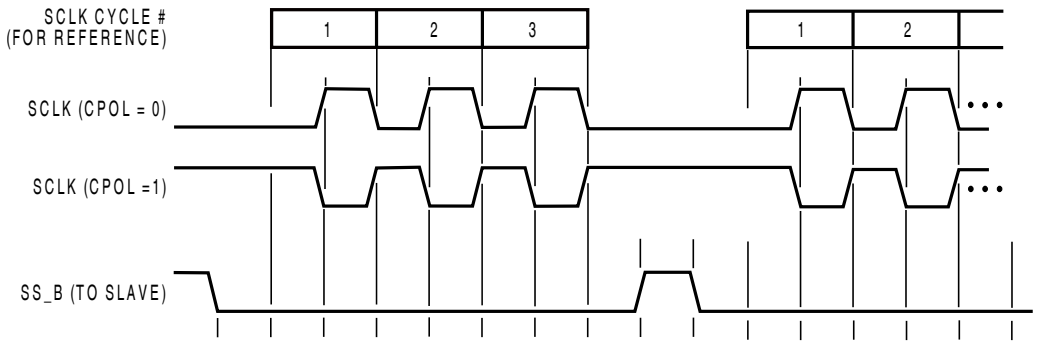


Figure 34-9. $\overline{SS}$ Strobe Timing (CPHA = 0)

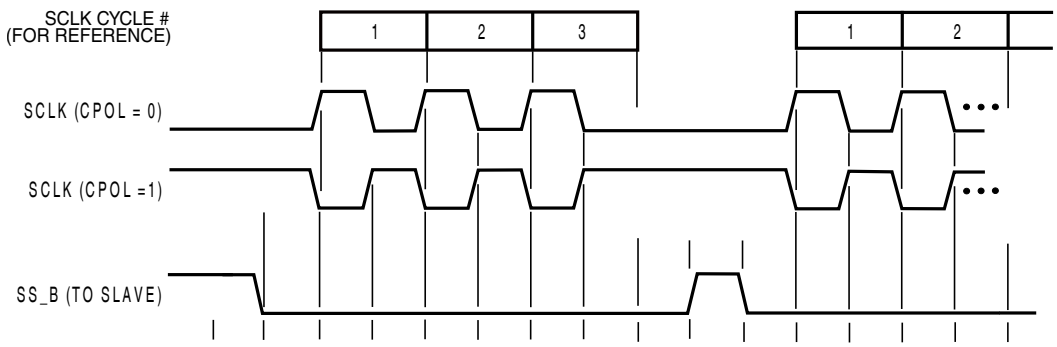


Figure 34-10. $\overline{SS}$ Strobe Timing (CPHA = 1)

If the SSB_AUTO bit is set in master mode, the SPI generates the initial falling edge and the final rising edge of $\overline{SS}$ for a slave device. The $\overline{SS}$ output has a falling edge one bit time before the first edge of SCLK (see the following figure).

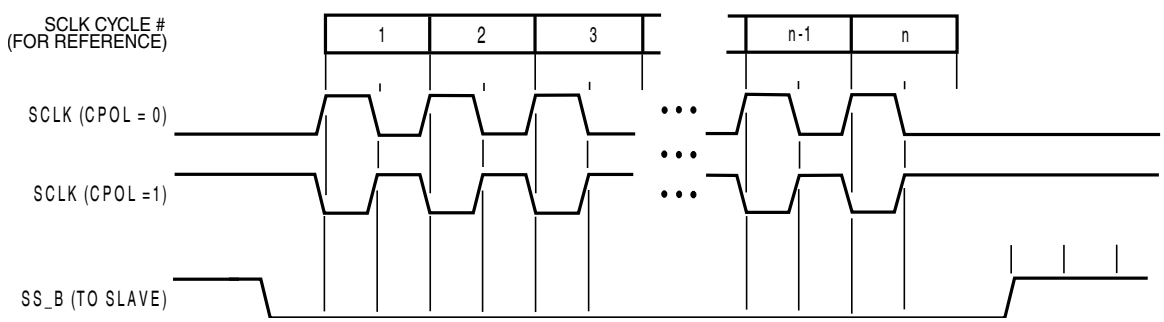


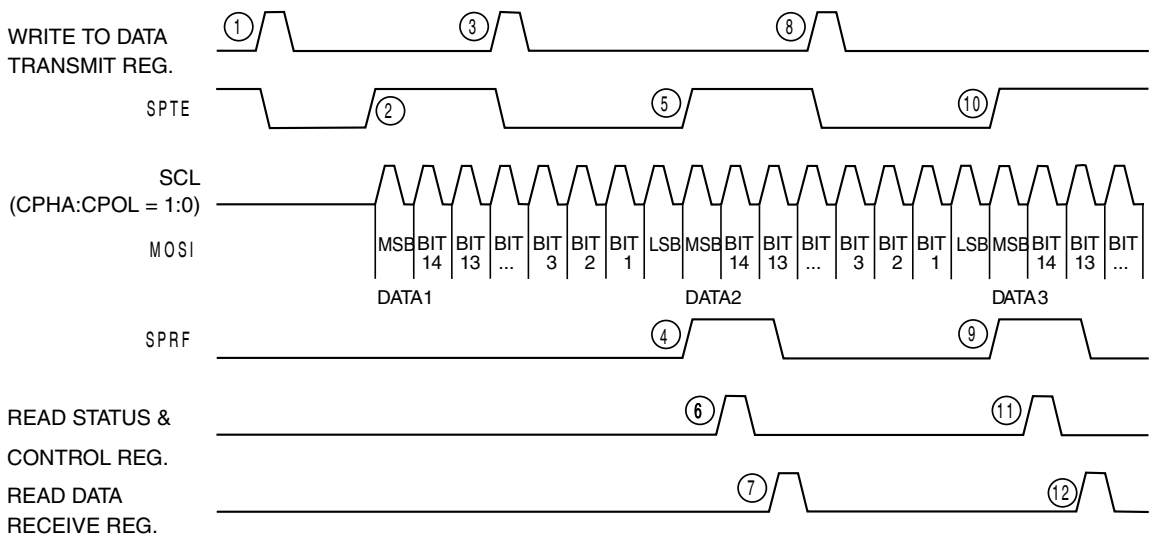
Figure 34-11. $\overline{SS}$ Auto Timing (CPHA = 1)

34.4.3 Transmission Data

The double-buffered data transmit register enables data to be queued and transmitted. For an SPI configured as a master, the queued data is transmitted immediately after the previous transaction has completed. The SPI transmitter empty flag (SPTE) indicates when the transmit data buffer is ready to accept new data. Write to the data transmit register only when the SPTE bit is high. The following figure shows the timing associated with doing back-to-back transactions with the SPI (SCLK has CPHA = 1; CPOL = 0).

Note

The following figure assumes 16-bit data lengths and the MSB shifted out first.



1. Device writes data 1 to Data Transmit register, clearing the SPTE bit.
2. Data 1 transfers from the Data Transmit register to the shift register, setting the SPTE bit.
3. Device writes data 2 to Data Transmit register, queueing data 2 and clearing the SPTE bit.
4. First incoming word transfers from shift register to the Data Receive register, settng the SPRF bit.
5. Data 2 transfers from the Data Transmit register to the shift register, setting the SPTE bit.
6. Device reads Status and Control register with the SPRF bit set.
7. Device reads Data Receive register, clearing the SPRF bit.
8. Device writes data 3 to Data Transmit register, queueing data 3 and clearing the SPTE bit.
9. The second incoming data transfers from the shift register to the Receive Data Register, setting the SPRF bit.
10. Data 3 transfers from the Data Transmit register to the shift register, setting the SPTE bit.
11. Device reads Status and Control register with the SPRF bit set.
12. Device reads Data Receive register, clearing the SPRF bit.

Figure 34-12. SPRF/SPTE Interrupt Timing

The transmit data buffer enables back-to-back transactions without the slave precisely timing its writes between transactions, as occurs in a system with a single data buffer. Also, in slave mode, if no new data is written to the SPI Data Transmit register, the last value contained in the SPI Data Transmit register is retransmitted if the external master starts a new transaction.

For an idle master that has no data loaded into its transmit buffer and no word currently being transmitted, the SPTE is set again no more than two bus cycles after the SPI Data Transmit register is written. This enables the user to queue up at most a 32-bit value to send. For an SPI operating in slave mode, the load of the shift register is controlled by the external master, and back-to-back writes to the transmit data register are not possible. The SPTE bit indicates when the next write can occur.

34.4.4 Error Conditions

The following flags signal SPI error conditions:

- Overflow (OVRF) — Failing to read the SPI Data Receive register before the next data word finishes entering the shift register sets the OVRF bit. The new data word does not transfer to the Receive Data register, and the unread data word can still be read. OVRF is in the SPI Status and Control register.
- Mode fault error (MODF) — The MODF bit indicates that the voltage on the slave select pin ($\overline{SS}$) is inconsistent with the mode of the SPI. MODF is in the SPI Status and Control register.

34.4.4.1 Overflow Error

The overflow flag (OVRF) is set if the SPI Receive Data register still has unread data from a previous transaction when the capture strobe of bit 1 of the next transaction occurs. The bit 1 capture strobe occurs in the middle of SCLK when the data length equals transaction data length minus 1. If an overflow occurs, all data received after the overflow and before the OVRF bit is cleared does not transfer to the SPI Receive Data register and does not set the SPI receiver full bit (SPRF). The unread data that is transferred to the SPI Receive Data register before the overflow occurred can still be read. Therefore, an overflow error always indicates the loss of data. Clear the overflow flag by reading the SPI Status and Control register and then reading the SPI Receive Data register.

OVRF generates a receiver/error interrupt request if the error interrupt enable bit (ERRIE) is also set. It is not possible to enable MODF or OVRF individually to generate a receiver/error interrupt request. However, leaving MODFEN low prevents MODF from being set.

If the SPRF interrupt is enabled and the ERRIE is not enabled, poll the OVRF bit to detect an overflow condition. The following figure shows how it is possible to miss an overflow. The first part of the figure shows how it is possible to read the SPI Status and Control register and the SPI Data Receive register to clear the SPRF without problems. However, as illustrated by the second transaction example, the OVRF bit can be set in between the time that the SPI Status and Control register and the SPI Data Receive register are read.

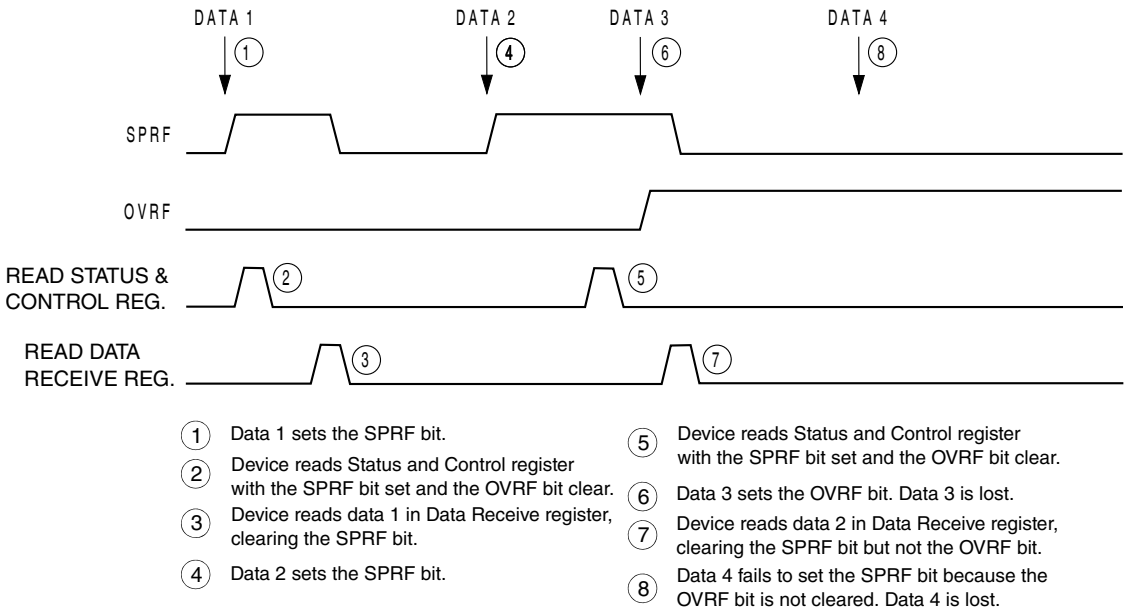


Figure 34-13. Missed Read of Overflow Condition

In this case, an overflow can easily be missed. Since no more SPRF interrupts can be generated until this OVRF is serviced, it is not obvious that data is being lost as more transactions are completed. To prevent this, either enable the OVRF interrupt or do another read of the SPI Status and Control register following the read of the SPI Data Receive register. This ensures that the OVRF was not set before the SPRF was cleared and that future transactions can set the SPRF bit. The following figure illustrates this process. Generally, to avoid this second read of the SPI Status and Control register, enable the OVRF to the device by setting the ERRIE bit.

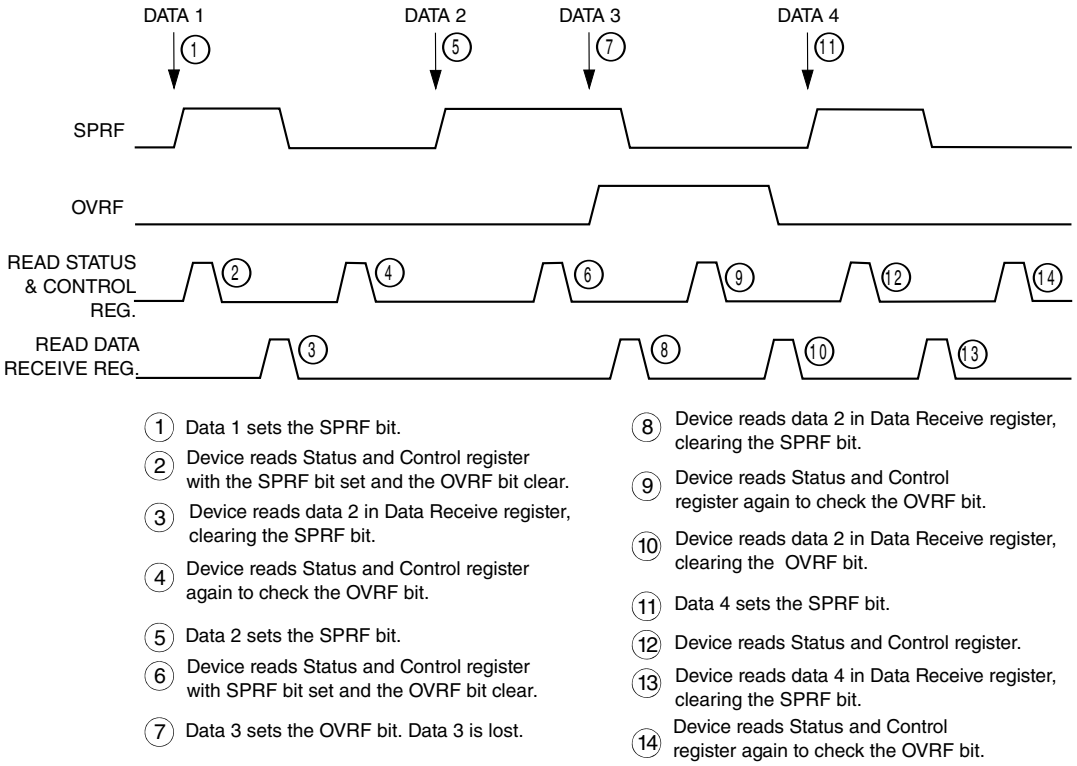


Figure 34-14. Clearing SPRF When OVRF Interrupt Is Not Enabled

34.4.4.2 Mode Fault Error

Setting the SPMSTR bit selects master mode and configures the SCLK and MOSI pins as outputs and the MISO pin as an input. Clearing SPMSTR selects slave mode and configures the SCLK and MOSI pins as inputs and the MISO pin as an output. The mode fault bit, MODF, is set any time the state of the slave select pin, $\overline{SS}$, is inconsistent with the mode selected by SPMSTR. To prevent SPI pin contention and damage to the device, a mode fault error occurs if:

- The $\overline{SS}$ pin of a slave SPI goes high during a transaction.
- The $\overline{SS}$ pin of a master SPI goes low at any time.

For the MODF flag to be set, the mode fault error enable bit (MODFEN) must be set. Clearing the MODFEN bit does not clear the MODF flag but does prevent MODF from being set again after MODF is cleared.

MODF generates a receiver/error interrupt request if the error interrupt enable bit (ERRIE) is also set. It is not possible to enable MODF or OVRF individually to generate a receiver/error interrupt request. However, leaving MODFEN low prevents MODF from being set.

34.4.4.2.1 Master Mode Fault

In a master SPI with the mode fault enable bit (MODFEN) set, the mode fault flag (MODF) is set if $\overline{SS}$ goes to logic zero. A mode fault in a master SPI causes the following events to occur:

- If ERRIE = 1, the SPI generates an SPI receiver/error interrupt request.
- The SPE bit is cleared (SPI disabled).
- The SPTE bit is set.
- The SPI state counter is cleared.

Note

Setting the MODF flag does not clear the SPMSTR bit. The SPMSTR bit has no function when SPE = 0. Reading SPMSTR when MODF = 1 shows the difference between a MODF occurring when the SPI is a master and when it is a slave.

In a master SPI, the MODF flag is not cleared until the $\overline{SS}$ pin is at a logic one or the SPI is configured as a slave.

34.4.4.2.2 Slave Mode Fault

When configured as a slave (SPMSTR = 0), the MODF flag is set if the $\overline{SS}$ pin goes high during a transaction. When CPHA = 0, a transaction begins when $\overline{SS}$ goes low and ends after the incoming SCLK goes back to its idle level following the shift of the last data bit. When CPHA = 1, the transaction begins when the SCLK leaves its idle level and $\overline{SS}$ is already low. The transaction continues until the SCLK returns to its idle level following the shift of the last data bit.

In a slave SPI (SPMSTR = 0), the MODF bit generates an SPI receiver/error interrupt request if the ERRIE bit is set. The MODF bit does not clear the SPE bit or reset the SPI in any way. Software can abort the SPI transaction by clearing the SPE bit of the slave.

Note

A logic one voltage on the $\overline{SS}$ pin of a slave SPI puts the MISO pin in a high impedance state. Also, the slave SPI ignores all incoming SCLK clocks, even if it was already in the middle of a transaction. A mode fault occurs if the $\overline{SS}$ pin changes state during a transaction.

When CPHA = 0, a MODF occurs if a slave is selected ($\overline{SS}$ is at logic 0) and later unselected ($\overline{SS}$ is at logic 1) after the first bit of data has been received (SCLK is toggled at least once). This happens because $\overline{SS}$ at logic 0 indicates the start of the transaction (MISO driven out with the value of MSB) for CPHA = 0. When CPHA = 1, a slave can be selected and then later unselected with no transaction occurring. Therefore, MODF does not occur because a transaction was never begun.

To clear the MODF flag, write a one to the MODF bit in the SPI Status and Control register. If the MODF flag is not cleared by writing a one to the MODF bit, the condition causing the mode fault still exists. In this case, the interrupt caused by the MODF flag can be cleared by disabling the EERIE bit or MODFEN bit (if set) or by disabling the SPI.

34.4.5 Resetting the SPI

Any system reset completely resets the SPI. Partial resets occur whenever the SPI enable bit (SPE) is low. Whenever SPE is low, the following occurs:

1. The SPTE flag is set.
2. Any slave mode transaction currently in progress is aborted.
3. Any master mode transaction currently in progress continues to completion.
4. The SPI state counter is cleared, making it ready for a new complete transaction.
5. All the SPI port logic is disabled.

Items 4 and 5 occur after 2 in slave mode, or after 3 in master mode.

The following items are reset only by a system reset:

- The SPI Data Transmit and SPI Data Receive registers
- All control bits in the SPI Status and Control register and the SPI Data Size and Control register
- The status flags SPRF, OVRF, and MODF

By not resetting the control bits when SPE is low, the user can clear SPE between transactions without having to set all control bits again when SPE is set back high for the next transaction.

By not resetting the SPRF, OVRF, and MODF flags, the user can still service these interrupts after the SPI is disabled. The user can disable the SPI by writing 0 to the SPE bit. The SPI is also disabled when a mode fault occurs in an SPI configured as a master.

34.5 Interrupts

Four SPI status flags can be enabled to generate device interrupt requests.

Table 34-3. SPI Interrupts

Flag	Interrupt Enabled By	Description
SPTE (Transmitter Empty)	SPI Enable / SPI Transmitter Interrupt Enable (SPTIE = 1, SPE = 1)	The SPI transmitter interrupt enable bit (SPTIE) enables the SPI transmitter empty (SPTE) flag or TFWM to generate transmitter interrupt requests, provided that the SPI is enabled (SPE = 1). The SPTE bit becomes set every time data transfers from the SPI Data Transmit register to the shift register and there is no more new data available in the TX queue. The clearing mechanism for the SPTE flag is a write to the SPI Data Transmit register.
SPRF (Receiver Full)	SPI Receiver Interrupt Enable (SPRIE = 1, SPE = 1)	The SPI receiver interrupt enable bit (SPRIE) enables the SPI receiver full (SPRF) bit or RFWM to generate receiver interrupt requests. The SPRF is set every time data transfers from the shift register to the SPI Data Receive register and there is no more room available in the RX queue to receive new data. The clearing mechanism for the SPRF flag is to read the SPI Data Receive register.
OVRF (Overflow)	SPI Receiver/Error Interrupt Enable (ERRIE = 1)	The error interrupt enable bit (ERRIE) enables both the MODF and OVRF bits to generate a receiver/error interrupt request.
MODF (Mode Fault)	SPI Receiver/Error Interrupt Enable (ERRIE = 1)	The mode fault enable bit (MODEFEN) enables the mode fault (MODF) bit to be set. The MODF bit allows the receiver/error interrupt request regardless of the state of the SPE bit as long as the interrupt enable (ERRIE) is set. The mode fault enable bit (MODFEN) can prevent the MODF flag from being set, so that only the OVRF bit is enabled by the ERRIE bit to generate receiver/error device interrupt requests.

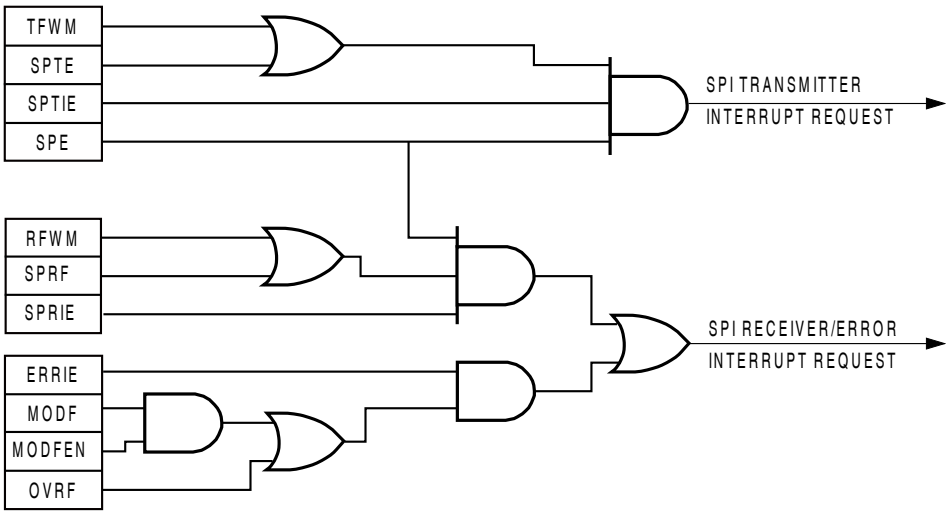


Figure 34-15. SPI Interrupt Request Generation

Chapter 35
Inter-Integrated Circuit (I2C)

35.1 Introduction

NOTE

For the chip-specific implementation details of this module's instances, see the chip configuration information.

The inter-integrated circuit (I^2C, I2C, or IIC) module provides a method of communication between a number of devices.

The interface is designed to operate up to 100 kbit/s with maximum bus loading and timing. The I2C 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. The I2C module also complies with the *System Management Bus (SMBus) Specification, version 2*.

35.1.1 Features

The I2C module has the following features:

- Compatible with *The I^2C-Bus Specification*
- Multimaster operation
- Software programmable for one of 64 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 and detection
- Repeated START signal generation and detection
- Acknowledge bit generation and detection

- Bus busy detection
- General call recognition
- 10-bit address extension
- Support for *System Management Bus (SMBus) Specification, version 2*
- Programmable glitch input filter
- Low power mode wakeup on slave address match
- Range slave address support
- DMA support

35.1.2 Modes of operation

The I2C module's operation in various low power modes is as follows:

- Run mode: This is the basic mode of operation. To conserve power in this mode, disable the module.
- Wait mode: The module continues to operate when the core is in Wait mode and can provide a wakeup interrupt.
- Stop mode: The module is inactive in Stop mode for reduced power consumption, except that address matching is enabled in Stop mode. The STOP instruction does not affect the I2C module's register states.

35.1.3 Block diagram

The following figure is a functional block diagram of the I2C module.

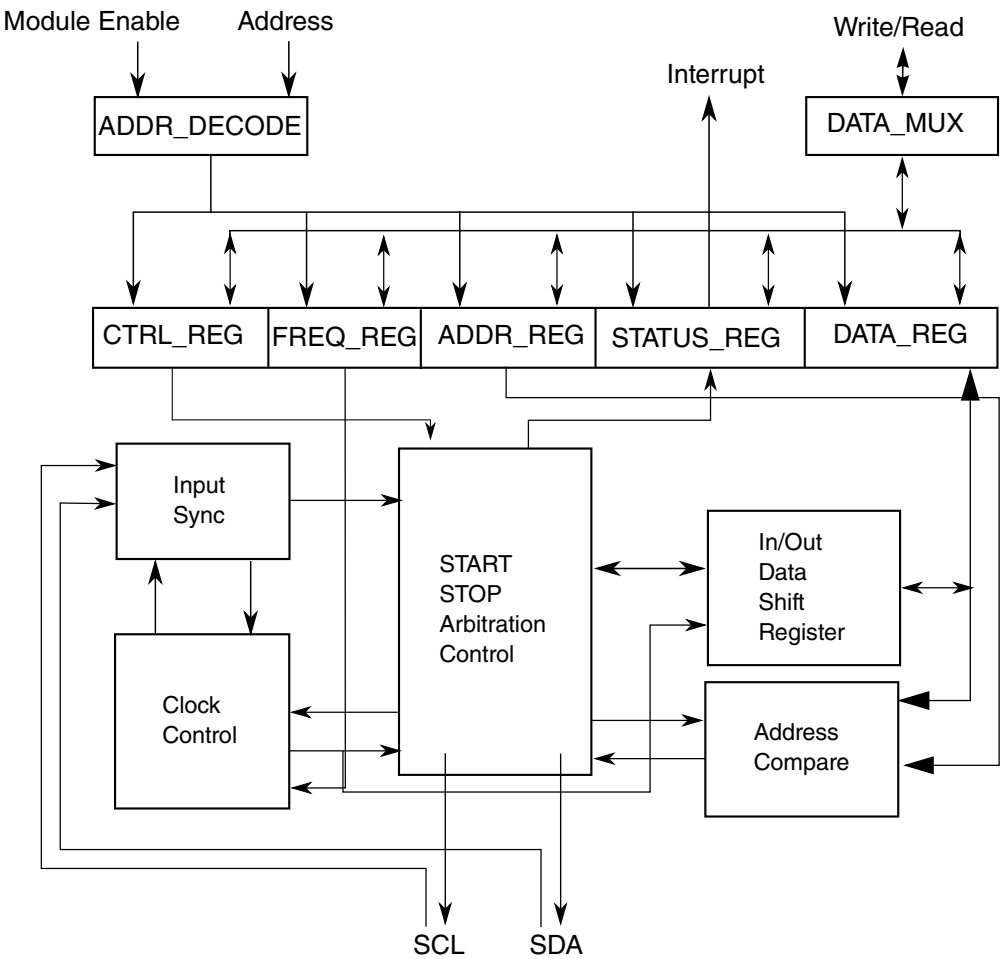


Figure 35-1. I2C Functional block diagram

35.2 I²C signal descriptions

The signal properties of I²C are shown in the table found here.

Table 35-1. I²C signal descriptions

Signal	Description	I/O
SCL	Bidirectional serial clock line of the I²C system.	I/O
SDA	Bidirectional serial data line of the I²C system.	I/O

35.3 Memory map/register definition

This section describes in detail all I2C registers accessible to the end user.

NOTE

Each 8-bit register occupies bits 0-7 of a 16-bit width. The other 8 bits are read-only and always read 0.

I2C memory map

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
E0E0	I2C Address Register 1 (I2C0_A1)	16	R/W	0000h	35.3.1/937
E0E1	I2C Frequency Divider register (I2C0_F)	16	R/W	0000h	35.3.2/937
E0E2	I2C Control Register 1 (I2C0_C1)	16	R/W	0000h	35.3.3/938
E0E3	I2C Status register (I2C0_S)	16	R/W	0080h	35.3.4/940
E0E4	I2C Data I/O register (I2C0_D)	16	R/W	0000h	35.3.5/942
E0E5	I2C Control Register 2 (I2C0_C2)	16	R/W	0000h	35.3.6/943
E0E6	I2C Programmable Input Glitch Filter register (I2C0_FLT)	16	R/W	0000h	35.3.7/944
E0E7	I2C Range Address register (I2C0_RA)	16	R/W	0000h	35.3.8/946
E0E8	I2C SMBus Control and Status register (I2C0_SMB)	16	R/W	0000h	35.3.9/946
E0E9	I2C Address Register 2 (I2C0_A2)	16	R/W	00C2h	35.3.10/ 948
E0EA	I2C SCL Low Timeout Register High (I2C0_SLTH)	16	R/W	0000h	35.3.11/ 948
E0EB	I2C SCL Low Timeout Register Low (I2C0_SLTL)	16	R/W	0000h	35.3.12/ 949
E0F0	I2C Address Register 1 (I2C1_A1)	16	R/W	0000h	35.3.1/937
E0F1	I2C Frequency Divider register (I2C1_F)	16	R/W	0000h	35.3.2/937
E0F2	I2C Control Register 1 (I2C1_C1)	16	R/W	0000h	35.3.3/938
E0F3	I2C Status register (I2C1_S)	16	R/W	0080h	35.3.4/940
E0F4	I2C Data I/O register (I2C1_D)	16	R/W	0000h	35.3.5/942
E0F5	I2C Control Register 2 (I2C1_C2)	16	R/W	0000h	35.3.6/943
E0F6	I2C Programmable Input Glitch Filter register (I2C1_FLT)	16	R/W	0000h	35.3.7/944
E0F7	I2C Range Address register (I2C1_RA)	16	R/W	0000h	35.3.8/946
E0F8	I2C SMBus Control and Status register (I2C1_SMB)	16	R/W	0000h	35.3.9/946
E0F9	I2C Address Register 2 (I2C1_A2)	16	R/W	00C2h	35.3.10/ 948
E0FA	I2C SCL Low Timeout Register High (I2C1_SLTH)	16	R/W	0000h	35.3.11/ 948
E0FB	I2C SCL Low Timeout Register Low (I2C1_SLTL)	16	R/W	0000h	35.3.12/ 949

35.3.1 I2C Address Register 1 (I2Cx_A1)

This register contains the slave address to be used by the I2C module.

Address: Base address + 0h offset

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	0								AD[7:1]							0
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

I2Cx_A1 field descriptions

Field	Description
15–8 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
7–1 AD[7:1]	Address Contains the primary slave address used by the I2C module when it is addressed as a slave. This field is used in the 7-bit address scheme and the lower seven bits in the 10-bit address scheme.
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

35.3.2 I2C Frequency Divider register (I2Cx_F)

Address: Base address + 1h offset

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	0								MULT		ICR					
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

I2Cx_F field descriptions

Field	Description
15–8 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
7–6 MULT	Multiplier Factor Defines the multiplier factor (mul). This factor is used along with the SCL divider to generate the I2C baud rate. 00 mul = 1 01 mul = 2 10 mul = 4 11 Reserved
ICR	ClockRate

Table continues on the next page...

I2Cx_F field descriptions (continued)

Field	Description

	Prescales the I2C module clock for bit rate selection. This field and the MULT field determine the I2C baud rate, the SDA hold time, the SCL start hold time, and the SCL stop hold time. For a list of values corresponding to each ICR setting, see I2C divider and hold values.
	The SCL divider multiplied by multiplier factor (mul) determines the I2C baud rate.
	`I2C baud rate = I2C module clock speed (Hz)/(mul × SCL divider)`
	The SDA hold time is the delay from the falling edge of SCL (I2C clock) to the changing of SDA (I2C data).
	`SDA hold time = I2C module clock period (s) × mul × SDA hold value`
	The SCL start hold time is the delay from the falling edge of SDA (I2C data) while SCL is high (start condition) to the falling edge of SCL (I2C clock).
	`SCL start hold time = I2C module clock period (s) × mul × SCL start hold value`
	The SCL stop hold time is the delay from the rising edge of SCL (I2C clock) to the rising edge of SDA (I2C data) while SCL is high (stop condition).
	`SCL stop hold time = I2C module clock period (s) × mul × SCL stop hold value`
	For example, if the I2C module clock speed is 8 MHz, the following table shows the possible hold time values with different ICR and MULT selections to achieve an I^2C baud rate of 100 kbit/s.

MULT	ICR	Hold times (µs)		
		SDA	SCL Start	SCL Stop
2h	00h	3.500	3.000	5.500
1h	07h	2.500	4.000	5.250
1h	0Bh	2.250	4.000	5.250
0h	14h	2.125	4.250	5.125
0h	18h	1.125	4.750	5.125

35.3.3 I2C Control Register 1 (I2Cx_C1)

Address: Base address + 2h offset

Bit	15	14	13	12	11	10	9	8
Read	0							
Write								
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
Read	IICEN	IICIE	MST	TX	TXAK	0	WUEN	DMAEN
Write						RSTA		
Reset	0	0	0	0	0	0	0	0

I2Cx_C1 field descriptions

Field	Description
15–8 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
7 IICEN	I2C Enable Enables I2C module operation. 0 Disabled 1 Enabled
6 IICIE	I2C Interrupt Enable Enables I2C interrupt requests. 0 Disabled 1 Enabled
5 MST	Master Mode Select When MST is changed from 0 to 1, a START signal is generated on the bus and master mode is selected. When this bit changes from 1 to 0, a STOP signal is generated and the mode of operation changes from master to slave. 0 Slave mode 1 Master mode
4 TX	Transmit Mode Select Selects the direction of master and slave transfers. In master mode this bit must be set according to the type of transfer required. Therefore, for address cycles, this bit is always set. When addressed as a slave this bit must be set by software according to the SRW bit in the status register. 0 Receive 1 Transmit
3 TXAK	Transmit Acknowledge Enable Specifies the value driven onto the SDA during data acknowledge cycles for both master and slave receivers. The value of SMB[FACK] affects NACK/ACK generation. **NOTE:** SCL is held low until TXAK is written. 0 An acknowledge signal is sent to the bus on the following receiving byte (if FACK is cleared) or the current receiving byte (if FACK is set). 1 No acknowledge signal is sent to the bus on the following receiving data byte (if FACK is cleared) or the current receiving data byte (if FACK is set).
2 RSTA	Repeat START Writing 1 to this bit generates a repeated START condition provided it is the current master. This bit will always be read as 0. Attempting a repeat at the wrong time results in loss of arbitration.
1 WUEN	Wakeup Enable The I2C module can wake the MCU from low power mode with no peripheral bus running when slave address matching occurs. 0 Normal operation. No interrupt generated when address matching in low power mode. 1 Enables the wakeup function in low power mode.

Table continues on the next page...

I2Cx_C1 field descriptions (continued)

Field	Description
0 DMAEN	DMA Enable Enables or disables the DMA function. 0 All DMA signalling disabled. 1 DMA transfer is enabled. While SMB[FACK] = 0, the following conditions trigger the DMA request: • a data byte is received, and either address or data is transmitted. (ACK/NACK is automatic) • the first byte received matches the A1 register or is a general call address. If any address matching occurs, S[IAAS] and S[TCF] are set. If the direction of transfer is known from master to slave, then it is not required to check S[SRW]. With this assumption, DMA can also be used in this case. In other cases, if the master reads data from the slave, then it is required to rewrite the C1 register operation. With this assumption, DMA cannot be used. When FACK = 1, an address or a data byte is transmitted.

35.3.4 I2C Status register (I2Cx_S)

Address: Base address + 3h offset

Bit	15	14	13	12	11	10	9	8
Read	0							
Write								
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
Read	TCF	IAAS	BUSY	ARBL	RAM	SRW	IICIF	RXAK
Write				w1c			w1c	
Reset	1	0	0	0	0	0	0	0

I2Cx_S field descriptions

Field	Description
15–8 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
7 TCF	Transfer Complete Flag Acknowledges a byte transfer; TCF sets on the completion of a byte transfer. This bit is valid only during or immediately following a transfer to or from the I2C module. TCF is cleared by reading the I2C data register in receive mode or by writing to the I2C data register in transmit mode. 0 Transfer in progress 1 Transfer complete
6 IAAS	Addressed As A Slave This bit is set by one of the following conditions:

Table continues on the next page...

I2Cx_S field descriptions (continued)

Field	Description
	• The calling address matches the programmed primary slave address in the A1 register, or matches the range address in the RA register (which must be set to a nonzero value and under the condition I2C_C2[RMEN] = 1). • C2[GCAEN] is set and a general call is received. • SMB[SIICAEN] is set and the calling address matches the second programmed slave address. • ALERTEN is set and an SMBus alert response address is received • RMEN is set and an address is received that is within the range between the values of the A1 and RA registers. IAAS sets before the ACK bit. The CPU must check the SRW bit and set TX/RX accordingly. Writing the C1 register with any value clears this bit. 0 Not addressed 1 Addressed as a slave
5 BUSY	Bus Busy Indicates the status of the bus regardless of slave or master mode. This bit is set when a START signal is detected and cleared when a STOP signal is detected. 0 Bus is idle 1 Bus is busy
4 ARBL	Arbitration Lost This bit is set by hardware when the arbitration procedure is lost. The ARBL bit must be cleared by software, by writing 1 to it. 0 Standard bus operation. 1 Loss of arbitration.
3 RAM	Range Address Match This bit is set to 1 by any of the following conditions, if I2C_C2[RMEN] = 1: • Any nonzero calling address is received that matches the address in the RA register. • The calling address is within the range of values of the A1 and RA registers. **NOTE:** For the RAM bit to be set to 1 correctly, C1[IICIE] must be set to 1. Writing the C1 register with any value clears this bit to 0. 0 Not addressed 1 Addressed as a slave
2 SRW	Slave Read/Write When addressed as a slave, SRW indicates the value of the R/$\overline{W}$ command bit of the calling address sent to the master. 0 Slave receive, master writing to slave 1 Slave transmit, master reading from slave
1 IICIF	Interrupt Flag This bit sets when an interrupt is pending. This bit must be cleared by software by writing 1 to it, such as in the interrupt routine. One of the following events can set this bit: • One byte transfer, including ACK/NACK bit, completes if FACK is 0. An ACK or NACK is sent on the bus by writing 0 or 1 to TXAK after this bit is set in receive mode. • One byte transfer, excluding ACK/NACK bit, completes if FACK is 1.

Table continues on the next page...

I2Cx_S field descriptions (continued)

Field	Description
	• Match of slave address to calling address including primary slave address, range slave address, alert response address, second slave address, or general call address. • Arbitration lost • In SMBus mode, any timeouts except SCL and SDA high timeouts • I2C bus stop or start detection if the SSIE bit in the Input Glitch Filter register is 1 **NOTE:** To clear the I2C bus stop or start detection interrupt: In the interrupt service routine, first clear the STOPF or STARTF bit in the Input Glitch Filter register by writing 1 to it, and then clear the IICIF bit. If this sequence is reversed, the IICIF bit is asserted again. 0 No interrupt pending 1 Interrupt pending
0 RXAK	Receive Acknowledge 0 Acknowledge signal was received after the completion of one byte of data transmission on the bus 1 No acknowledge signal detected

35.3.5 I2C Data I/O register (I2Cx_D)

Address: Base address + 4h offset

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	0								DATA							
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

I2Cx_D field descriptions

Field	Description
15–8 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
DATA	Data In master transmit mode, when data is written to this register, a data transfer is initiated. The most significant bit is sent first. In master receive mode, reading this register initiates receiving of the next byte of data. **NOTE:** When making the transition out of master receive mode, switch the I2C mode before reading the Data register to prevent an inadvertent initiation of a master receive data transfer. In slave mode, the same functions are available after an address match occurs. The C1[TX] bit must correctly reflect the desired direction of transfer in master and slave modes for the transmission to begin. For example, if the I2C module is configured for master transmit but a master receive is desired, reading the Data register does not initiate the receive. Reading the Data register returns the last byte received while the I2C module is configured in master receive or slave receive mode. The Data register does not reflect every byte that is transmitted on the I2C bus, and neither can software verify that a byte has been written to the Data register correctly by reading it back.

Table continues on the next page...

I2Cx_D field descriptions (continued)

Field	Description
	In master transmit mode, the first byte of data written to the Data register following assertion of MST (start bit) or assertion of RSTA (repeated start bit) is used for the address transfer and must consist of the calling address (in bits 7-1) concatenated with the required R/$\overline{W}$ bit (in position bit 0).

35.3.6 I2C Control Register 2 (I2Cx_C2)

Address: Base address + 5h offset

Bit	15	14	13	12	11	10	9	8
Read	0							
Write								
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
Read Write	GCAEN	ADEXT	HDRS	SBRC	RMEN	AD[10:8]		
Reset	0	0	0	0	0	0	0	0

I2Cx_C2 field descriptions

Field	Description
15–8 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
7 GCAEN	General Call Address Enable Enables general call address. 0 Disabled 1 Enabled
6 ADEXT	Address Extension Controls the number of bits used for the slave address. 0 7-bit address scheme 1 10-bit address scheme
5 HDRS	High Drive Select Controls the drive capability of the I2C pads. 0 Normal drive mode 1 High drive mode
4 SBRC	Slave Baud Rate Control Enables independent slave mode baud rate at maximum frequency, which forces clock stretching on SCL in very fast I2C modes. To a slave, an example of a "very fast" mode is when the master transfers at 40 kbit/s but the slave can capture the master's data at only 10 kbit/s. 0 The slave baud rate follows the master baud rate and clock stretching may occur 1 Slave baud rate is independent of the master baud rate

Table continues on the next page...

I2Cx_C2 field descriptions (continued)

Field	Description
3 RMEN	Range Address Matching Enable This bit controls the slave address matching for addresses between the values of the A1 and RA registers. When this bit is set, a slave address matching occurs for any address greater than the value of the A1 register and less than or equal to the value of the RA register. 0 Range mode disabled. No address matching occurs for an address within the range of values of the A1 and RA registers. 1 Range mode enabled. Address matching occurs when a slave receives an address within the range of values of the A1 and RA registers.
AD[10:8]	Slave Address Contains the upper three bits of the slave address in the 10-bit address scheme. This field is valid only while the ADEXT bit is set.

35.3.7 I2C Programmable Input Glitch Filter register (I2Cx_FLT)

Address: Base address + 6h offset

Bit	15	14	13	12	11	10	9	8
Read	0							
Write								
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
Read	SHEN	STOPF	SSIE	STARTF	0	FLT		
Write		w1c		w1c				
Reset	0	0	0	0	0	0	0	0

I2Cx_FLT field descriptions

Field	Description
15–8 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
7 SHEN	Stop Hold Enable Set this bit to hold off entry to stop mode when any data transmission or reception is occurring. The following scenario explains the holdoff functionality: 1. The I2C module is configured for a basic transfer, and the SHEN bit is set to 1. 2. A transfer begins. 3. The MCU signals the I2C module to enter stop mode. 4. The byte currently being transferred, including both address and data, completes its transfer. 5. The I2C slave or master acknowledges that the in-transfer byte completed its transfer and acknowledges the request to enter stop mode. 6. After receiving the I2C module's acknowledgment of the request to enter stop mode, the MCU determines whether to shut off the I2C module's clock.

Table continues on the next page...

I2Cx_FLT field descriptions (continued)

Field	Description
	If the SHEN bit is set to 1 and the I2C module is in an idle or disabled state when the MCU signals to enter stop mode, the module immediately acknowledges the request to enter stop mode. If SHEN is cleared to 0 and the overall data transmission or reception that was suspended by stop mode entry was incomplete: To resume the overall transmission or reception after the MCU exits stop mode, software must reinitialize the transfer by resending the address of the slave. If the I2C Control Register 1's IICIE bit was set to 1 before the MCU entered stop mode, system software will receive the interrupt triggered by the I2C Status Register's TCF bit after the MCU wakes from the stop mode. 0 Stop holdoff is disabled. The MCU's entry to stop mode is not gated. 1 Stop holdoff is enabled.
6 STOPF	I2C Bus Stop Detect Flag Hardware sets this bit when the I2C bus's stop status is detected. The STOPF bit must be cleared by writing 1 to it. 0 No stop happens on I2C bus 1 Stop detected on I2C bus
5 SSIE	I2C Bus Stop or Start Interrupt Enable This bit enables the interrupt for I2C bus stop or start detection. **NOTE:** To clear the I2C bus stop or start detection interrupt: In the interrupt service routine, first clear the STOPF or STARTF bit by writing 1 to it, and then clear the IICIF bit in the status register. If this sequence is reversed, the IICIF bit is asserted again. 0 Stop or start detection interrupt is disabled 1 Stop or start detection interrupt is enabled
4 STARTF	I2C Bus Start Detect Flag Hardware sets this bit when the I2C bus's start status is detected. The STARTF bit must be cleared by writing 1 to it. 0 No start happens on I2C bus 1 Start detected on I2C bus
3 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
FLT	I2C Programmable Filter Factor Controls the width of the glitch, in terms of I2C module clock cycles, that the filter must absorb. For any glitch whose size is less than or equal to this width setting, the filter does not allow the glitch to pass. 0h No filter/bypass 1-Fh Filter glitches up to width of *n* I2C module clock cycles, where *n*=1-7d

35.3.8 I2C Range Address register (I2C*x*_RA)

Address: Base address + 7h offset

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	0								RAD							0
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

I2C*x*_RA field descriptions

Field	Description
15–8 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
7–1 RAD	Range Slave Address This field contains the slave address to be used by the I2C module. The field is used in the 7-bit address scheme. If I2C_C2[RMEN] is set to 1, any nonzero value write enables this register. This register value can be considered as a maximum boundary in the range matching mode.
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

35.3.9 I2C SMBus Control and Status register (I2C*x*_SMB)

NOTE

When the SCL and SDA signals are held high for a length of time greater than the high timeout period, the SHTF1 flag sets. Before reaching this threshold, while the system is detecting how long these signals are being held high, a master assumes that the bus is free. However, the SHTF1 bit is set to 1 in the bus transmission process with the idle bus state.

NOTE

When the TCKSEL bit is set, there is no need to monitor the SHTF1 bit because the bus speed is too high to match the protocol of SMBus.

Address: Base address + 8h offset

Bit	15	14	13	12	11	10	9	8
Read	0							
Write								
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
Read	FACK	ALERTEN	SIICAEN	TCKSEL	SLTF	SHTF1	SHTF2	SHTF2IE
Write					w1c		w1c	
Reset	0	0	0	0	0	0	0	0

I2Cx_SMB field descriptions

Field	Description
15–8 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
7 FACK	Fast NACK/ACK Enable For SMBus packet error checking, the CPU must be able to issue an ACK or NACK according to the result of receiving data byte. 0 An ACK or NACK is sent on the following receiving data byte 1 Writing 0 to TXAK after receiving a data byte generates an ACK. Writing 1 to TXAK after receiving a data byte generates a NACK.
6 ALERTEN	SMBus Alert Response Address Enable Enables or disables SMBus alert response address matching. **NOTE:** After the host responds to a device that used the alert response address, you must use software to put the device's address on the bus. The alert protocol is described in the SMBus specification. 0 SMBus alert response address matching is disabled 1 SMBus alert response address matching is enabled
5 SIICAEN	Second I2C Address Enable Enables or disables SMBus device default address. 0 I2C address register 2 matching is disabled 1 I2C address register 2 matching is enabled
4 TCKSEL	Timeout Counter Clock Select Selects the clock source of the timeout counter. 0 Timeout counter counts at the frequency of the I2C module clock / 64 1 Timeout counter counts at the frequency of the I2C module clock
3 SLTF	SCL Low Timeout Flag This bit is set when the SLT register (consisting of the SLTH and SLTL registers) is loaded with a non-zero value (LoValue) and an SCL low timeout occurs. Software clears this bit by writing a logic 1 to it. **NOTE:** The low timeout function is disabled when the SLT register's value is 0. 0 No low timeout occurs 1 Low timeout occurs
2 SHTF1	SCL High Timeout Flag 1 This read-only bit sets when SCL and SDA are held high more than clock × LoValue / 512, which indicates the bus is free. This bit is cleared automatically.

Table continues on the next page...

I2Cx_SMB field descriptions (continued)

Field	Description
	0 No SCL high and SDA high timeout occurs 1 SCL high and SDA high timeout occurs
1 SHTF2	SCL High Timeout Flag 2 This bit sets when SCL is held high and SDA is held low more than clock × LoValue / 512. Software clears this bit by writing 1 to it. 0 No SCL high and SDA low timeout occurs 1 SCL high and SDA low timeout occurs
0 SHTF2IE	SHTF2 Interrupt Enable Enables SCL high and SDA low timeout interrupt. 0 SHTF2 interrupt is disabled 1 SHTF2 interrupt is enabled

35.3.10 I2C Address Register 2 (I2Cx_A2)

Address: Base address + 9h offset

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	0								SAD							0
Write																
Reset	0	0	0	0	0	0	0	0	1	1	0	0	0	0	1	0

I2Cx_A2 field descriptions

Field	Description
15–8 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
7–1 SAD	SMBus Address Contains the slave address used by the SMBus. This field is used on the device default address or other related addresses.
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

35.3.11 I2C SCL Low Timeout Register High (I2Cx_SLTH)

Address: Base address + Ah offset

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	0								SSLT[15:8]							
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

I2Cx_SLTH field descriptions

Field	Description
15–8 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
SSLT[15:8]	SSLT[15:8] Most significant byte of SCL low timeout value that determines the timeout period of SCL low.

35.3.12 I2C SCL Low Timeout Register Low (I2Cx_SLTL)

Address: Base address + Bh offset

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	0								SSLT[7:0]							
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

I2Cx_SLTL field descriptions

Field	Description
15–8 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
SSLT[7:0]	SSLT[7:0] Least significant byte of SCL low timeout value that determines the timeout period of SCL low.

35.4 Functional description

This section provides a comprehensive functional description of the I2C module.

35.4.1 I2C protocol

The I2C bus system uses a serial data line (SDA) and a serial clock line (SCL) for data transfers.

All devices connected to it must have open drain or open collector outputs. A logic AND function is exercised on both lines with external pull-up resistors. The value of these resistors depends on the system.

Normally, a standard instance of communication is composed of four parts:

1. START signal
2. Slave address transmission

3. Data transfer
4. STOP signal

The STOP signal should not be confused with the CPU STOP instruction. The following figure illustrates I2C bus system communication.

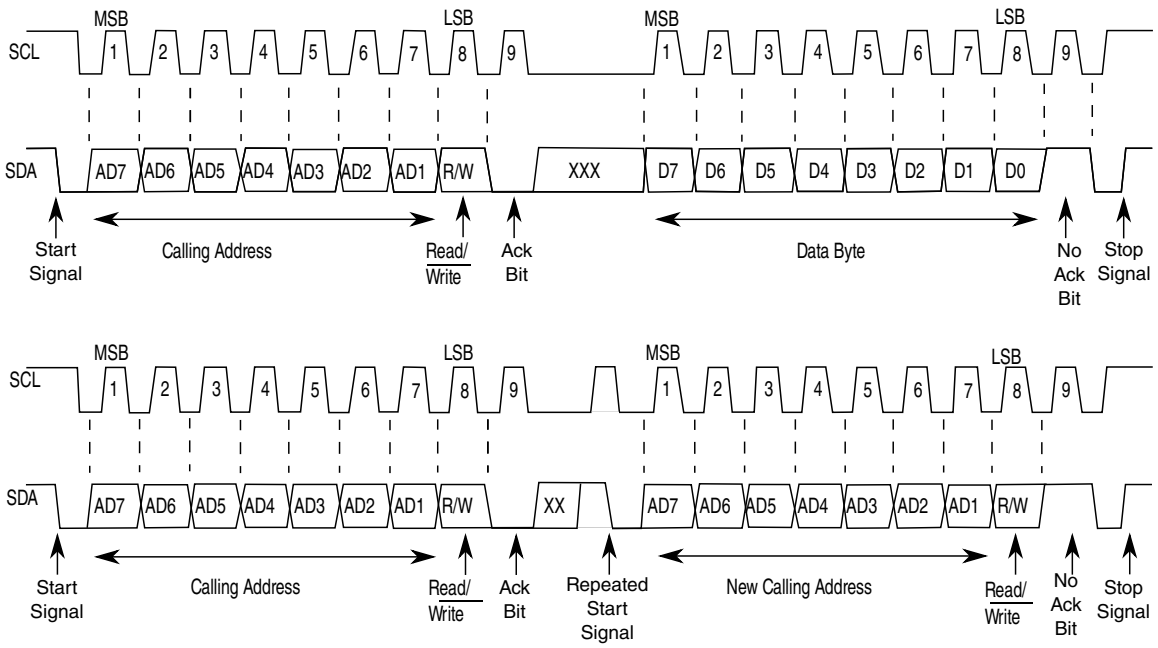


Figure 35-2. I2C bus transmission signals

35.4.1.1 START signal

The bus is free when no master device is engaging the bus (both SCL and SDA are high). When the bus is free, a master may initiate communication by sending a START signal. 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 might contain several bytes of data—and brings all slaves out of their idle states.

35.4.1.2 Slave address transmission

Immediately after the START signal, the first byte of a data transfer is the slave address transmitted by the master. This address is a 7-bit calling address followed by an R/$\overline{W}$ bit. The R/$\overline{W}$ 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

Only the slave with a calling address that matches the one transmitted by the master responds by sending an acknowledge bit. The slave sends the acknowledge bit by pulling SDA low at the ninth clock.

No two slaves in the system can have the same address. If the I2C module is the master, it must not transmit an address that is equal to its own slave address. The I2C module cannot be master and slave at the same time. However, if arbitration is lost during an address cycle, the I2C module reverts to slave mode and operates correctly even if it is being addressed by another master.

35.4.1.3 Data transfers

When successful slave addressing is achieved, data transfer can proceed on a byte-by-byte basis in the direction specified by the R/$\overline{W}$ bit sent by the calling master.

All transfers that follow an address cycle are referred to as data transfers, even if they carry subaddress information for the slave device.

Each data byte is 8 bits long. Data may be changed only while SCL is low. Data must be held stable while SCL is high. There is one clock pulse on SCL for each data bit, and the MSB is transferred first. Each data byte is followed by a ninth (acknowledge) bit, which is signaled from the receiving device by pulling SDA low at the ninth clock. In summary, one complete data transfer needs nine clock pulses.

If the slave receiver does not acknowledge the master in the ninth bit, the slave must leave SDA high. The master interprets the failed acknowledgement as an unsuccessful data transfer.

If the master receiver does not acknowledge the slave transmitter after a data byte transmission, the slave interprets it as an end to data transfer and releases the SDA line.

In the case of a failed acknowledgement by either the slave or master, the data transfer is aborted and the master does one of two things:

- Relinquishes the bus by generating a STOP signal.
- Commences a new call by generating a repeated START signal.

35.4.1.4 STOP signal

The master can terminate the communication by generating a STOP signal to free the bus. A STOP signal is defined as a low-to-high transition of SDA while SCL is asserted.

The master can generate a STOP signal even if the slave has generated an acknowledgement, at which point the slave must release the bus.

35.4.1.5 Repeated START signal

The master may generate a START signal followed by a calling command without generating a STOP signal first. This action is called a repeated START. The master uses a repeated START to communicate with another slave or with the same slave in a different mode (transmit/receive mode) without releasing the bus.

35.4.1.6 Arbitration procedure

The I2C bus is a true multimaster 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. The bus clock's low period is equal to the longest clock low period, and the high period 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 level 1 while another master transmits logic level 0. The losing masters immediately switch 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, hardware sets a status bit to indicate the loss of arbitration.

35.4.1.7 Clock synchronization

Because wire AND logic is performed on SCL, a high-to-low transition on SCL affects all devices connected on the bus. The devices start counting their low period and, after a device's clock has gone low, that device holds SCL low until the clock reaches its high state. However, the change of low to high in this device clock might not change the state of SCL if another device clock is still within its low period. Therefore, the 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 the following diagram. When all applicable devices have counted off their low period, the synchronized clock SCL is released and pulled high. Afterward there is no difference between the device clocks and the state of SCL, and all devices start counting their high periods. The first device to complete its high period pulls SCL low again.

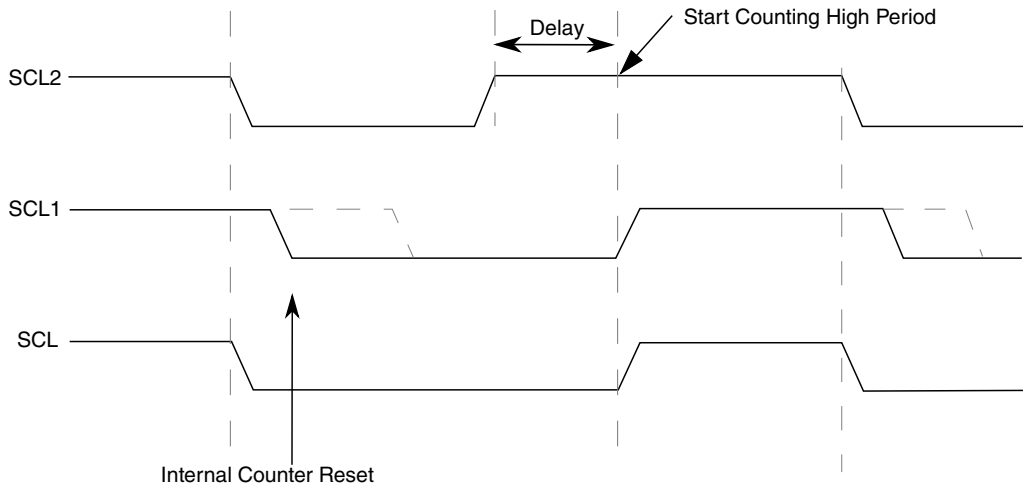


Figure 35-3. I2C clock synchronization

35.4.1.8 Handshaking

The clock synchronization mechanism can be used as a handshake in data transfers. A slave device may hold SCL low after completing a single byte transfer (9 bits). In this case, it halts the bus clock and forces the master clock into wait states until the slave releases SCL.

35.4.1.9 Clock stretching

The clock synchronization mechanism can be used by slaves to slow down the bit rate of a transfer. After the master drives SCL low, a slave can drive SCL low for the required period and then release it. If the slave's SCL low period is greater than the master's SCL low period, the resulting SCL bus signal's low period is stretched. In other words, the SCL bus signal's low period is increased to be the same length as the slave's SCL low period.

35.4.1.10 I2C divider and hold values

NOTE

For some cases on some devices, the SCL divider value may vary by ±2 or ±4 when ICR's value ranges from 00h to 0Fh. These potentially varying SCL divider values are highlighted in the following table. For the actual SCL divider values for your device, see the chip-specific details about the I2C module.

Table 35-2. I2C divider and hold values

ICR (hex)	SCL divider	SDA hold value	SCL hold (start) value	SCL hold (stop) value	ICR (hex)	SCL divider (clocks)	SDA hold (clocks)	SCL hold (start) value	SCL hold (stop) value
00	20	7	6	11	20	160	17	78	81
01	22	7	7	12	21	192	17	94	97
02	24	8	8	13	22	224	33	110	113
03	26	8	9	14	23	256	33	126	129
04	28	9	10	15	24	288	49	142	145
05	30	9	11	16	25	320	49	158	161
06	34	10	13	18	26	384	65	190	193
07	40	10	16	21	27	480	65	238	241
08	28	7	10	15	28	320	33	158	161
09	32	7	12	17	29	384	33	190	193
0A	36	9	14	19	2A	448	65	222	225
0B	40	9	16	21	2B	512	65	254	257
0C	44	11	18	23	2C	576	97	286	289
0D	48	11	20	25	2D	640	97	318	321
0E	56	13	24	29	2E	768	129	382	385
0F	68	13	30	35	2F	960	129	478	481
10	48	9	18	25	30	640	65	318	321
11	56	9	22	29	31	768	65	382	385
12	64	13	26	33	32	896	129	446	449
13	72	13	30	37	33	1024	129	510	513
14	80	17	34	41	34	1152	193	574	577
15	88	17	38	45	35	1280	193	638	641
16	104	21	46	53	36	1536	257	766	769
17	128	21	58	65	37	1920	257	958	961
18	80	9	38	41	38	1280	129	638	641
19	96	9	46	49	39	1536	129	766	769
1A	112	17	54	57	3A	1792	257	894	897
1B	128	17	62	65	3B	2048	257	1022	1025
1C	144	25	70	73	3C	2304	385	1150	1153
1D	160	25	78	81	3D	2560	385	1278	1281
1E	192	33	94	97	3E	3072	513	1534	1537
1F	240	33	118	121	3F	3840	513	1918	1921

35.4.2 10-bit address

For 10-bit addressing, 0x11110 is used for the first 5 bits of the first address byte. Various combinations of read/write formats are possible within a transfer that includes 10-bit addressing.

35.4.2.1 Master-transmitter addresses a slave-receiver

The transfer direction is not changed. When a 10-bit address follows a START condition, each slave compares the first 7 bits of the first byte of the slave address (11110XX) with its own address and tests whether the eighth bit (R/$\overline{W}$ direction bit) is 0. It is possible that more than one device finds a match and generates an acknowledge (A1). Each slave that finds a match compares the 8 bits of the second byte of the slave address with its own address, but only one slave finds a match and generates an acknowledge (A2). The matching slave remains addressed by the master until it receives a STOP condition (P) or a repeated START condition (Sr) followed by a different slave address.

Table 35-3. Master-transmitter addresses slave-receiver with a 10-bit address

S	Slave address first 7 bits 11110 + AD10 + AD9	R/$\overline{W}$ 0	A1	Slave address second byte AD[8:1]	A2	Data	A	...	Data	A/A	P

After the master-transmitter has sent the first byte of the 10-bit address, the slave-receiver sees an I2C interrupt. User software must ensure that for this interrupt, the contents of the Data register are ignored and not treated as valid data.

35.4.2.2 Master-receiver addresses a slave-transmitter

The transfer direction is changed after the second R/$\overline{W}$ bit. Up to and including acknowledge bit A2, the procedure is the same as that described for a master-transmitter addressing a slave-receiver. After the repeated START condition (Sr), a matching slave remembers that it was addressed before. This slave then checks whether the first seven bits of the first byte of the slave address following Sr are the same as they were after the START condition (S), and it tests whether the eighth (R/$\overline{W}$) bit is 1. If there is a match, the slave considers that it has been addressed as a transmitter and generates acknowledge A3. The slave-transmitter remains addressed until it receives a STOP condition (P) or a repeated START condition (Sr) followed by a different slave address.

After a repeated START condition (Sr), all other slave devices also compare the first seven bits of the first byte of the slave address with their own addresses and test the eighth (R/$\overline{W}$) bit. However, none of them are addressed because R/$\overline{W}$ = 1 (for 10-bit devices), or the 11110XX slave address (for 7-bit devices) does not match.

Table 35-4. Master-receiver addresses a slave-transmitter with a 10-bit address

S	Slave address first 7 bits 11110 + AD10 + AD9	R/$\overline{W}$ 0	A1	Slave address second byte AD[8:1]	A2	Sr	Slave address first 7 bits 11110 + AD10 + AD9	R/$\overline{W}$ 1	A3	Data	A	...	Data	A	P

After the master-receiver has sent the first byte of the 10-bit address, the slave-transmitter sees an I2C interrupt. User software must ensure that for this interrupt, the contents of the Data register are ignored and not treated as valid data.

35.4.3 Address matching

All received addresses can be requested in 7-bit or 10-bit address format.

- AD[7:1] in Address Register 1, which contains the I2C primary slave address, always participates in the address matching process. It provides a 7-bit address.
- If the ADEXT bit is set, AD[10:8] in Control Register 2 participates in the address matching process. It extends the I2C primary slave address to a 10-bit address.

Additional conditions that affect address matching include:

- If the GCAEN bit is set, general call participates the address matching process.
- If the ALERTEN bit is set, alert response participates the address matching process.
- If the SIICAEN bit is set, Address Register 2 participates in the address matching process.
- If the RMEN bit is set, when the Range Address register is programmed to a nonzero value, any address within the range of values of Address Register 1 (excluded) and the Range Address register (included) participates in the address matching process. The Range Address register must be programmed to a value greater than the value of Address Register 1.

When the I2C module responds to one of these addresses, it acts as a slave-receiver and the IAAS bit is set after the address cycle. Software must read the Data register after the first byte transfer to determine that the address is matched.

35.4.4 System management bus specification

SMBus provides a control bus for system and power management related tasks. A system can use SMBus to pass messages to and from devices instead of tripping individual control lines.

Removing the individual control lines reduces pin count. Accepting messages ensures future expandability. With the system management bus, a device can provide manufacturer information, tell the system what its model/part number is, save its state for a suspend event, report different types of errors, accept control parameters, and return its status.

35.4.4.1 Timeouts

The $T_{TIMEOUT,MIN}$ parameter allows a master or slave to conclude that a defective device is holding the clock low indefinitely or a master is intentionally trying to drive devices off the bus. The slave device must release the bus (stop driving the bus and let SCL and SDA float high) when it detects any single clock held low longer than $T_{TIMEOUT,MIN}$. Devices that have detected this condition must reset their communication and be able to receive a new START condition within the timeframe of $T_{TIMEOUT,MAX}$.

SMBus defines a clock low timeout, $T_{TIMEOUT}$, of 35 ms, specifies $T_{LOW:SEXT}$ as the cumulative clock low extend time for a slave device, and specifies $T_{LOW:MEXT}$ as the cumulative clock low extend time for a master device.

35.4.4.1.1 SCL low timeout

If the SCL line is held low by a slave device on the bus, no further communication is possible. Furthermore, the master cannot force the SCL line high to correct the error condition. To solve this problem, the SMBus protocol specifies that devices participating in a transfer must detect any clock cycle held low longer than a timeout value condition. Devices that have detected the timeout condition must reset the communication. When the I2C module is an active master, if it detects that SMBCLK low has exceeded the value of $T_{TIMEOUT,MIN}$, it must generate a stop condition within or after the current data byte in the transfer process. When the I2C module is a slave, if it detects the $T_{TIMEOUT,MIN}$ condition, it resets its communication and is then able to receive a new START condition.

35.4.4.1.2 SCL high timeout

When the I2C module has determined that the SMBCLK and SMBDAT signals have been high for at least $T_{HIGH:MAX}$, it assumes that the bus is idle.

A HIGH timeout occurs after a START condition appears on the bus but before a STOP condition appears on the bus. Any master detecting this scenario can assume the bus is free when either of the following occurs:

- SHTF1 rises.
- The BUSY bit is high and SHTF1 is high.

When the SMBDAT signal is low and the SMBCLK signal is high for a period of time, another kind of timeout occurs. The time period must be defined in software. SHTF2 is used as the flag when the time limit is reached. This flag is also an interrupt resource, so it triggers IICIF.

35.4.4.1.3 CSMBCLK TIMEOUT MEXT and CSMBCLK TIMEOUT SEXT

The following figure illustrates the definition of the timeout intervals $T_{LOW:SEXT}$ and $T_{LOW:MEXT}$. When in master mode, the I2C module must not cumulatively extend its clock cycles for a period greater than $T_{LOW:MEXT}$ within a byte, where each byte is defined as START-to-ACK, ACK-to-ACK, or ACK-to-STOP. When CSMBCLK TIMEOUT MEXT occurs, SMBus MEXT rises and also triggers the SLTF.

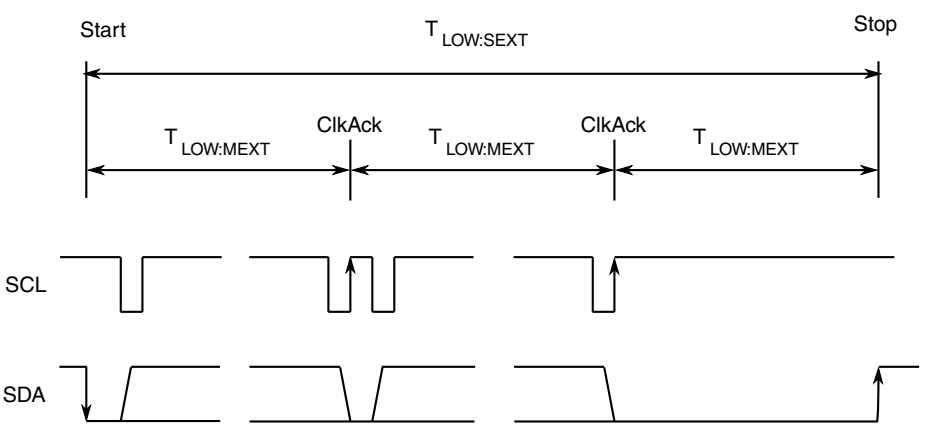


Figure 35-4. Timeout measurement intervals

A master is allowed to abort the transaction in progress to any slave that violates the $T_{LOW:SEXT}$ or $T_{TIMEOUT,MIN}$ specifications. To abort the transaction, the master issues a STOP condition at the conclusion of the byte transfer in progress. When a slave, the I2C module must not cumulatively extend its clock cycles for a period greater than $T_{LOW:SEXT}$ during any message from the initial START to the STOP. When CSMBCLK TIMEOUT SEXT occurs, SEXT rises and also triggers SLTF.

NOTE

CSMBCLK TIMEOUT SEXT and CSMBCLK TIMEOUT MEXT are optional functions that are implemented in the second step.

35.4.4.2 FAST ACK and NACK

To improve reliability and communication robustness, implementation of packet error checking (PEC) by SMBus devices is optional for SMBus devices but required for devices participating in and only during the address resolution protocol (ARP) process. The PEC is a CRC-8 error checking byte, calculated on all the message bytes. The PEC is appended to the message by the device that supplied the last data byte. If the PEC is present but not correct, a NACK is issued by the receiver. Otherwise an ACK is issued. To calculate the CRC-8 by software, this module can hold the SCL line low after receiving the eighth SCL (8th bit) if this byte is a data byte. So software can determine whether an ACK or NACK should be sent to the bus by setting or clearing the TXAK bit if the FACK (fast ACK/NACK enable) bit is enabled.

SMBus requires a device always to acknowledge its own address, as a mechanism to detect the presence of a removable device (such as a battery or docking station) on the bus. In addition to indicating a slave device busy condition, SMBus uses the NACK mechanism to indicate the reception of an invalid command or invalid data. Because such a condition may occur on the last byte of the transfer, SMBus devices are required to have the ability to generate the not acknowledge after the transfer of each byte and before the completion of the transaction. This requirement is important because SMBus does not provide any other resend signaling. This difference in the use of the NACK signaling has implications on the specific implementation of the SMBus port, especially in devices that handle critical system data such as the SMBus host and the SBS components.

NOTE

In the last byte of master receive slave transmit mode, the master must send a NACK to the bus, so FACK must be switched off before the last byte transmits.

35.4.5 Resets

The I2C module is disabled after a reset. The I2C module cannot cause a core reset.

35.4.6 Interrupts

The I2C module generates an interrupt when any of the events in the table found here occur, provided that the IICIE bit is set.

The interrupt is driven by the IICIF bit (of the I2C Status Register) and masked with the IICIE bit (of the I2C Control Register 1). The IICIF bit must be cleared (by software) by writing 1 to it in the interrupt routine. The SMBus timeouts interrupt is driven by SLTF and masked with the IICIE bit. The SLTF bit must be cleared by software by writing 1 to it in the interrupt routine. You can determine the interrupt type by reading the Status Register.

NOTE

In master receive mode, the FACK bit must be set to zero before the last byte transfer.

Table 35-5. Interrupt summary

Interrupt source	Status	Flag	Local enable
Complete 1-byte transfer	TCF	IICIF	IICIE
Match of received calling address	IAAS	IICIF	IICIE
Arbitration lost	ARBL	IICIF	IICIE
I^2C bus stop detection	STOPF	IICIF	IICIE & SSIE
I^2C bus start detection	STARTF	IICIF	IICIE & SSIE
SMBus SCL low timeout	SLTF	IICIF	IICIE
SMBus SCL high SDA low timeout	SHTF2	IICIF	IICIE & SHTF2IE
Wakeup from stop or wait mode	IAAS	IICIF	IICIE & WUEN

35.4.6.1 Byte transfer interrupt

The Transfer Complete Flag (TCF) bit is set at the falling edge of the ninth clock to indicate the completion of a byte and acknowledgement transfer. When FACK is enabled, TCF is then set at the falling edge of eighth clock to indicate the completion of byte.

35.4.6.2 Address detect interrupt

When the calling address matches the programmed slave address (I2C Address Register) or when the GCAEN bit is set and a general call is received, the IAAS bit in the Status Register is set. The CPU is interrupted, provided the IICIE bit is set. The CPU must check the SRW bit and set its Tx mode accordingly.

35.4.6.3 Stop Detect Interrupt

When the stop status is detected on the I^2C bus, the STOPF bit is set to 1. The CPU is interrupted, provided the IICIE and STOPIE bits are both set to 1.

35.4.6.4 Exit from low-power/stop modes

The slave receive input detect circuit and address matching feature are still active on low power modes (wait and stop). An asynchronous input matching slave address or general call address brings the CPU out of low power/stop mode if the interrupt is not masked. Therefore, TCF and IAAS both can trigger this interrupt.

35.4.6.5 Arbitration lost interrupt

The I2C is a true multimaster 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, the relative priority of the contending masters is determined by a data arbitration procedure. The I2C module asserts the arbitration-lost interrupt when it loses the data arbitration process and the ARBL bit in the Status Register is set.

Arbitration is lost in the following circumstances:

1. SDA is sampled as low when the master drives high during an address or data transmit cycle.
2. SDA is sampled as low when the master drives high during the acknowledge bit of a data receive cycle.
3. A START cycle is attempted when the bus is busy.
4. A repeated START cycle is requested in slave mode.
5. A STOP condition is detected when the master did not request it.

The ARBL bit must be cleared (by software) by writing 1 to it.

35.4.6.6 Timeout interrupt in SMBus

When the IICIE bit is set, the I2C module asserts a timeout interrupt (outputs SLTF and SHTF2) upon detection of any of the mentioned timeout conditions, with one exception. The SCL high and SDA high TIMEOUT mechanism must not be used to influence the timeout interrupt output, because this timeout indicates an idle condition on the bus. SHTF1 rises when it matches the SCL high and SDA high TIMEOUT and falls automatically just to indicate the bus status. The SHTF2's timeout period is the same as that of SHTF1, which is short compared to that of SLTF, so another control bit, SHTF2IE, is added to enable or disable it.

35.4.7 Programmable input glitch filter

An I2C glitch filter has been added outside legacy I2C modules but within the I2C package. This filter can absorb glitches on the I2C clock and data lines for the I2C module.

The width of the glitch to absorb can be specified in terms of the number of (half) I2C module clock cycles. A single Programmable Input Glitch Filter control register is provided. Effectively, any down-up-down or up-down-up transition on the data line that occurs within the number of clock cycles programmed in this register is ignored by the I2C module. The programmer must specify the size of the glitch (in terms of I2C module clock cycles) for the filter to absorb and not pass.

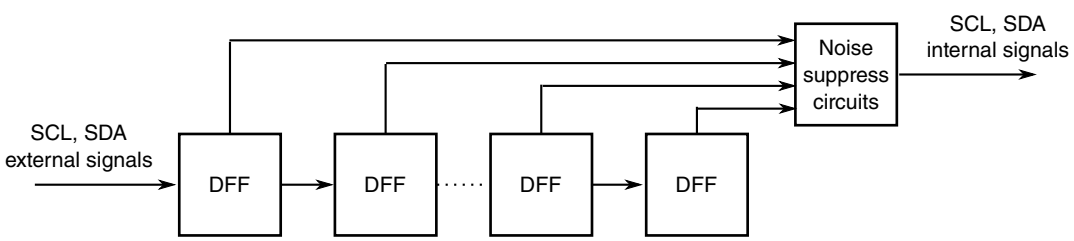


Figure 35-5. Programmable input glitch filter diagram

35.4.8 Address matching wake-up

When a primary, range, or general call address match occurs when the I2C module is in slave receive mode, the MCU wakes from a low power mode where no peripheral bus is running.

Data sent on the bus that is the same as a target device address might also wake the target MCU.

After the address matching IAAS bit is set, an interrupt is sent at the end of address matching to wake the core. The IAAS bit must be cleared after the clock recovery.

NOTE

After the system recovers and is in Run mode, restart the I2C module if it is needed to transfer packets. To avoid I2C transfer problems resulting from the situation, firmware should prevent the MCU execution of a STOP instruction when the I2C module is in the middle of a transfer unless the Stop mode holdoff feature is used during this period (set FLT[SHEN] to 1).

35.4.9 DMA support

If the DMAEN bit is cleared and the IICIE bit is set, an interrupt condition generates an interrupt request.

If the DMAEN bit is set and the IICIE bit is set, an interrupt condition generates a DMA request instead. DMA requests are generated by the transfer complete flag (TCF).

If the DMAEN bit is set, only the TCF initiates a DMA request. All other events generate CPU interrupts.

NOTE

Before the last byte of master receive mode, TXAK must be set to send a NACK after the last byte's transfer. Therefore, the DMA must be disabled before the last byte's transfer.

NOTE

In 10-bit address mode transmission, the addresses to send occupy 2–3 bytes. During this transfer period, the DMA must be disabled because the C1 register is written to send a repeat start or to change the transfer direction.

35.5 Initialization/application information

Module Initialization (Slave)

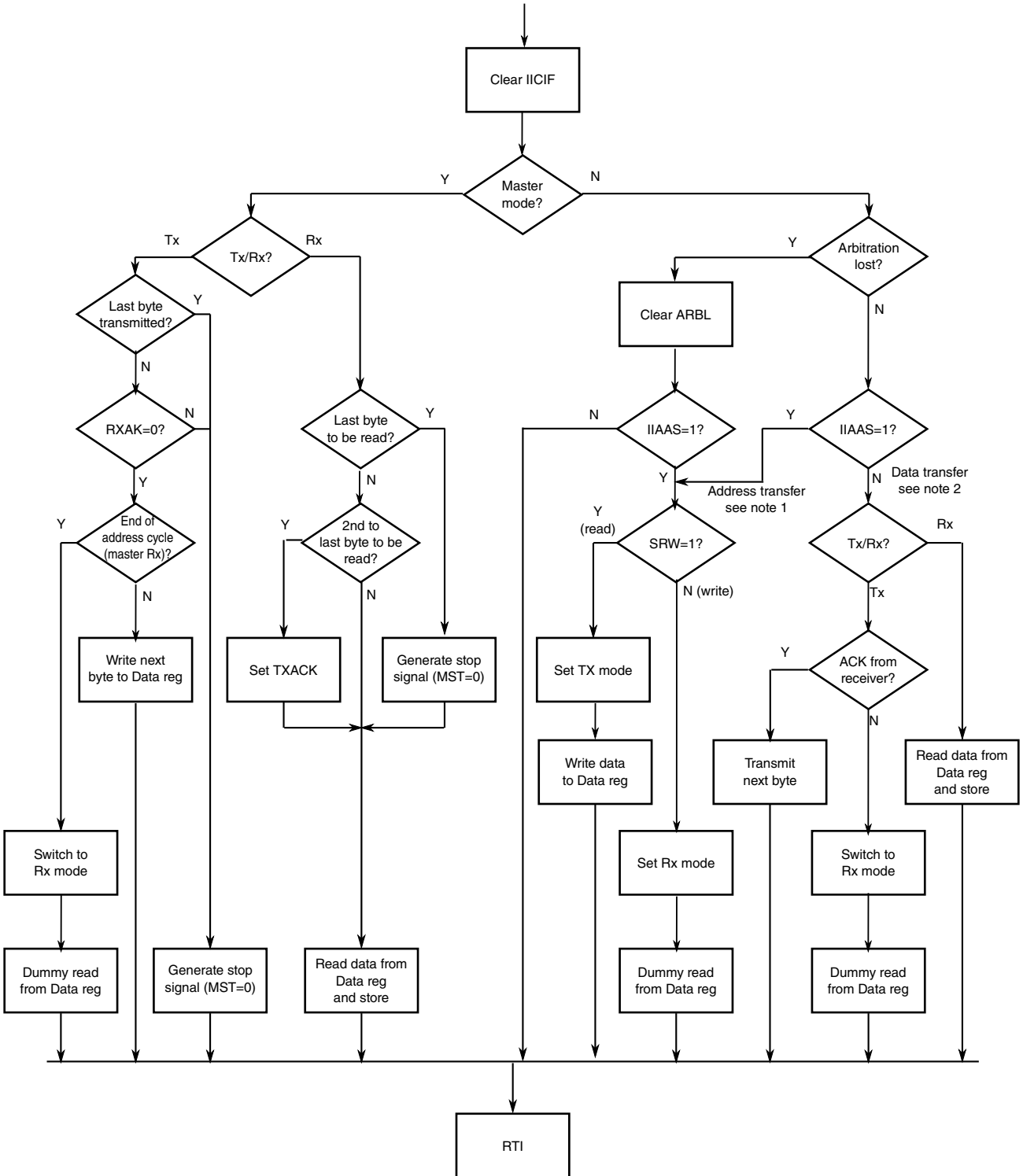
1. Write: Control Register 2
   - to enable or disable general call
   - to select 10-bit or 7-bit addressing mode
2. Write: Address Register 1 to set the slave address

3. Write: Control Register 1 to enable the I2C module and interrupts
4. Initialize RAM variables (IICEN = 1 and IICIE = 1) for transmit data
5. Initialize RAM variables used to achieve the routine shown in the following figure

Module Initialization (Master)

1. Write: Frequency Divider register to set the I2C baud rate (see example in description of ICR)
2. Write: Control Register 1 to enable the I2C module and interrupts
3. Initialize RAM variables (IICEN = 1 and IICIE = 1) for transmit data
4. Initialize RAM variables used to achieve the routine shown in the following figure
5. Write: Control Register 1 to enable TX
6. Write: Control Register 1 to enable MST (master mode)
7. Write: Data register with the address of the target slave (the LSB of this byte determines whether the communication is master receive or transmit)

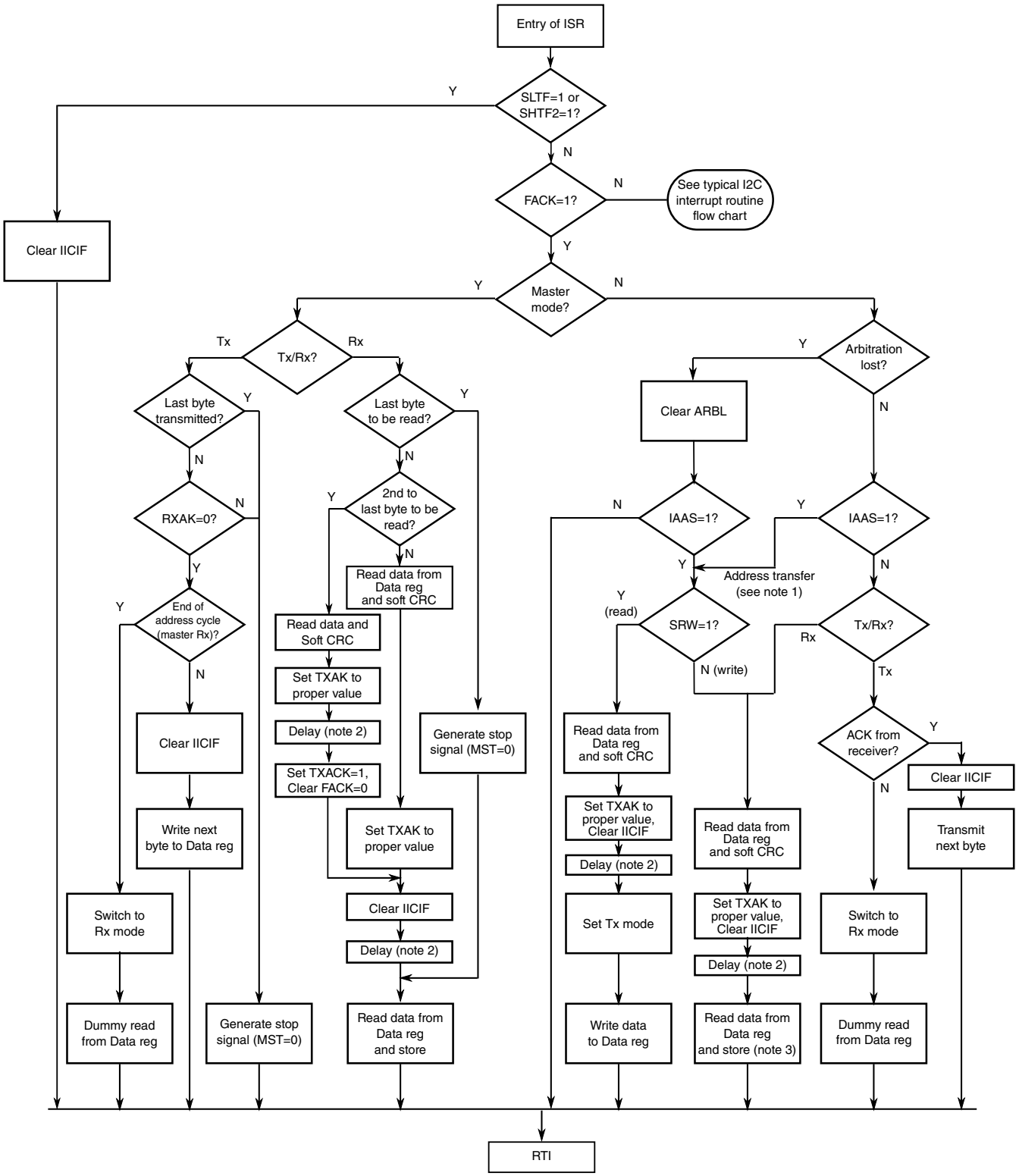
The routine shown in the following figure encompasses both master and slave I2C operations. For slave operation, an incoming I2C message that contains the proper address begins I2C communication. For master operation, communication must be initiated by writing the Data register. An example of an I2C driver which implements many of the steps described here is available in AN4342: Using the Inter-Integrated Circuit on ColdFire+ and Kinetis .



Notes:

1. If general call is enabled, check to determine if the received address is a general call address (0x00). If the received address is a general call address, the general call must be handled by user software.
2. When 10-bit addressing addresses a slave, the slave sees an interrupt following the first byte of the extended address. Ensure that for this interrupt, the contents of the Data register are ignored and not treated as a valid data transfer.

Figure 35-6. Typical I2C interrupt routine



Notes:

1. If general call or SIICAEN is enabled, check to determine if the received address is a general call address (0x00) or an SMBus device default address. In either case, they must be handled by user software.
2. In receive mode, one bit time delay may be needed before the first and second data reading, to wait for the possible longest time period (in worst case) of the 9th SCL cycle.
3. This read is a dummy read in order to reset the SMBus receiver state machine.

Figure 35-7. Typical I2C SMBus interrupt routine

Chapter 36
General-Purpose Input/Output (GPIO)

36.1 Overview

The general-purpose input/output (GPIO) module allows direct read or write access to pin values or the ability to assign a pin to be used as an external interrupt. GPIO pins are multiplexed with other peripherals on the package. The device's data sheet specifies the assigned GPIO ports and the multiplexed pin package.

A GPIO pin can be configured in different operation modes:

- As GPIO input with, or without, pull resistor
- As GPIO output with push-pull mode or open-drain mode
- As a peripheral pin when multiplexed with another module

GPIOs are placed on the device in groups of one to sixteen bits, called ports and designated as A, B, C, and so on. Refer to the device's data sheet for the specific definition of each of the GPIO ports on the chip.

36.1.1 Features

The GPIO module's design includes these features:

- Individual control for each pin to be in either peripheral mode or GPIO mode
- Individual direction control for each pin in GPIO mode
- Individual pull resistor enable control for each pin in either peripheral mode or GPIO mode
- Individual pull resistor type selection for each pin in either peripheral mode or GPIO mode
- Individual selection of output push-pull mode or open-drain mode for each pin
- Individual output drive strength control (high-power mode or low-power mode) for each pin
- Ability to monitor each pin's logic values when pin is in either GPIO mode or peripheral mode by using raw data (RDATA) register

- Each pin has the ability to generate an interrupt with programmable rising or falling edge and software interrupt
- 5 V tolerance
- Output edge slew rate control for each pin to reduce switch noise

36.1.2 Modes of Operation

The GPIO module can operate in two major modes:

- Peripheral mode: The peripheral module controls the pin. However, if the pin is not configured as an analog input, then output drive strength, edge slew rate control, push-pull or open-drain output, and pull resistor enable and type select remain controlled by GPIO registers.
- GPIO mode: The GPIO module controls the pin. Any data output and input can be written to or read from GPIO data registers. Pull resistor enables and type select are controlled by a GPIO register. GPIO pins can generate the edge interrupt and insert the software interrupt.

36.2 Memory Map and Registers

Each GPIO register contains up to 16 bits, each of which performs an identical function for one of the GPIO pins controlled by that GPIO port. However, initial operating conditions at reset can vary; some GPIO modes are on by default, and others are not.

NOTE

The reset value of these registers may differ depending on the reset function of specific pins. Refer to the device's data sheet.

For simplicity, each GPIO port's registers appear with the same width of 16 bits, corresponding to 16 pins. The actual number of pins per port (and therefore the number of usable control bits per port register) is as follows:

- Port A: 12
- Port B: 12
- Port C: 16
- Port D: 8
- Port E: 10
- Port F: 16

Any register bit for which no corresponding port pin exists, such as bits 15-8 of every GPIOD register, is reserved and always reads as 0.

GPIO memory map

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
E200	GPIO Pull Resistor Enable Register (GPIOA_PUR)	16	R/W	See section	36.2.1/971
E201	GPIO Data Register (GPIOA_DR)	16	R/W	Undefined	36.2.2/972
E202	GPIO Data Direction Register (GPIOA_DDR)	16	R/W	0000h	36.2.3/973
E203	GPIO Peripheral Enable Register (GPIOA_PER)	16	R/W	See section	36.2.4/973
E204	GPIO Interrupt Assert Register (GPIOA_IAR)	16	R/W	0000h	36.2.5/974
E205	GPIO Interrupt Enable Register (GPIOA_IENR)	16	R/W	0000h	36.2.6/974
E206	GPIO Interrupt Polarity Register (GPIOA_IPOLR)	16	R/W	0000h	36.2.7/975
E207	GPIO Interrupt Pending Register (GPIOA_IPR)	16	R	0000h	36.2.8/975
E208	GPIO Interrupt Edge Sensitive Register (GPIOA_IESR)	16	R/W	0000h	36.2.9/976
E209	GPIO Push-Pull Mode Register (GPIOA_PPMODE)	16	R/W	See section	36.2.10/ 976
E20A	GPIO Raw Data Register (GPIOA_RAWDATA)	16	R	Undefined	36.2.11/ 977
E20B	GPIO Drive Strength Control Register (GPIOA_DRIVE)	16	R/W	See section	36.2.12/ 978
E20C	GPIO Pull Resistor Type Select (GPIOA_PUS)	16	R/W	See section	36.2.13/ 978
E20D	Slew Rate Control Register (GPIOA_SRE)	16	R/W	See section	36.2.14/ 979
E210	GPIO Pull Resistor Enable Register (GPIOB_PUR)	16	R/W	See section	36.2.1/971
E211	GPIO Data Register (GPIOB_DR)	16	R/W	Undefined	36.2.2/972
E212	GPIO Data Direction Register (GPIOB_DDR)	16	R/W	0000h	36.2.3/973
E213	GPIO Peripheral Enable Register (GPIOB_PER)	16	R/W	See section	36.2.4/973
E214	GPIO Interrupt Assert Register (GPIOB_IAR)	16	R/W	0000h	36.2.5/974
E215	GPIO Interrupt Enable Register (GPIOB_IENR)	16	R/W	0000h	36.2.6/974
E216	GPIO Interrupt Polarity Register (GPIOB_IPOLR)	16	R/W	0000h	36.2.7/975
E217	GPIO Interrupt Pending Register (GPIOB_IPR)	16	R	0000h	36.2.8/975
E218	GPIO Interrupt Edge Sensitive Register (GPIOB_IESR)	16	R/W	0000h	36.2.9/976
E219	GPIO Push-Pull Mode Register (GPIOB_PPMODE)	16	R/W	See section	36.2.10/ 976
E21A	GPIO Raw Data Register (GPIOB_RAWDATA)	16	R	Undefined	36.2.11/ 977
E21B	GPIO Drive Strength Control Register (GPIOB_DRIVE)	16	R/W	See section	36.2.12/ 978
E21C	GPIO Pull Resistor Type Select (GPIOB_PUS)	16	R/W	See section	36.2.13/ 978
E21D	Slew Rate Control Register (GPIOB_SRE)	16	R/W	See section	36.2.14/ 979
E220	GPIO Pull Resistor Enable Register (GPIOC_PUR)	16	R/W	See section	36.2.1/971
E221	GPIO Data Register (GPIOC_DR)	16	R/W	Undefined	36.2.2/972
E222	GPIO Data Direction Register (GPIOC_DDR)	16	R/W	0000h	36.2.3/973

Table continues on the next page...

GPIO memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
E223	GPIO Peripheral Enable Register (GPIOC_PER)	16	R/W	See section	36.2.4/973
E224	GPIO Interrupt Assert Register (GPIOC_IAR)	16	R/W	0000h	36.2.5/974
E225	GPIO Interrupt Enable Register (GPIOC_IENR)	16	R/W	0000h	36.2.6/974
E226	GPIO Interrupt Polarity Register (GPIOC_IPOLR)	16	R/W	0000h	36.2.7/975
E227	GPIO Interrupt Pending Register (GPIOC_IPR)	16	R	0000h	36.2.8/975
E228	GPIO Interrupt Edge Sensitive Register (GPIOC_IESR)	16	R/W	0000h	36.2.9/976
E229	GPIO Push-Pull Mode Register (GPIOC_PPMODE)	16	R/W	See section	36.2.10/ 976
E22A	GPIO Raw Data Register (GPIOC_RAWDATA)	16	R	Undefined	36.2.11/ 977
E22B	GPIO Drive Strength Control Register (GPIOC_DRIVE)	16	R/W	See section	36.2.12/ 978
E22C	GPIO Pull Resistor Type Select (GPIOC_PUS)	16	R/W	See section	36.2.13/ 978
E22D	Slew Rate Control Register (GPIOC_SRE)	16	R/W	See section	36.2.14/ 979
E230	GPIO Pull Resistor Enable Register (GPIOD_PUR)	16	R/W	See section	36.2.1/971
E231	GPIO Data Register (GPIOD_DR)	16	R/W	Undefined	36.2.2/972
E232	GPIO Data Direction Register (GPIOD_DDR)	16	R/W	0000h	36.2.3/973
E233	GPIO Peripheral Enable Register (GPIOD_PER)	16	R/W	See section	36.2.4/973
E234	GPIO Interrupt Assert Register (GPIOD_IAR)	16	R/W	0000h	36.2.5/974
E235	GPIO Interrupt Enable Register (GPIOD_IENR)	16	R/W	0000h	36.2.6/974
E236	GPIO Interrupt Polarity Register (GPIOD_IPOLR)	16	R/W	0000h	36.2.7/975
E237	GPIO Interrupt Pending Register (GPIOD_IPR)	16	R	0000h	36.2.8/975
E238	GPIO Interrupt Edge Sensitive Register (GPIOD_IESR)	16	R/W	0000h	36.2.9/976
E239	GPIO Push-Pull Mode Register (GPIOD_PPMODE)	16	R/W	See section	36.2.10/ 976
E23A	GPIO Raw Data Register (GPIOD_RAWDATA)	16	R	Undefined	36.2.11/ 977
E23B	GPIO Drive Strength Control Register (GPIOD_DRIVE)	16	R/W	See section	36.2.12/ 978
E23C	GPIO Pull Resistor Type Select (GPIOD_PUS)	16	R/W	See section	36.2.13/ 978
E23D	Slew Rate Control Register (GPIOD_SRE)	16	R/W	See section	36.2.14/ 979
E240	GPIO Pull Resistor Enable Register (GPIOE_PUR)	16	R/W	See section	36.2.1/971
E241	GPIO Data Register (GPIOE_DR)	16	R/W	Undefined	36.2.2/972
E242	GPIO Data Direction Register (GPIOE_DDR)	16	R/W	0000h	36.2.3/973
E243	GPIO Peripheral Enable Register (GPIOE_PER)	16	R/W	See section	36.2.4/973
E244	GPIO Interrupt Assert Register (GPIOE_IAR)	16	R/W	0000h	36.2.5/974
E245	GPIO Interrupt Enable Register (GPIOE_IENR)	16	R/W	0000h	36.2.6/974

Table continues on the next page...

GPIO memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
E246	GPIO Interrupt Polarity Register (GPIOE_IPOLR)	16	R/W	0000h	36.2.7/975
E247	GPIO Interrupt Pending Register (GPIOE_IPR)	16	R	0000h	36.2.8/975
E248	GPIO Interrupt Edge Sensitive Register (GPIOE_IESR)	16	R/W	0000h	36.2.9/976
E249	GPIO Push-Pull Mode Register (GPIOE_PPMODE)	16	R/W	See section	36.2.10/ 976
E24A	GPIO Raw Data Register (GPIOE_RAWDATA)	16	R	Undefined	36.2.11/ 977
E24B	GPIO Drive Strength Control Register (GPIOE_DRIVE)	16	R/W	See section	36.2.12/ 978
E24C	GPIO Pull Resistor Type Select (GPIOE_PUS)	16	R/W	See section	36.2.13/ 978
E24D	Slew Rate Control Register (GPIOE_SRE)	16	R/W	See section	36.2.14/ 979
E250	GPIO Pull Resistor Enable Register (GPIOF_PUR)	16	R/W	See section	36.2.1/971
E251	GPIO Data Register (GPIOF_DR)	16	R/W	Undefined	36.2.2/972
E252	GPIO Data Direction Register (GPIOF_DDR)	16	R/W	0000h	36.2.3/973
E253	GPIO Peripheral Enable Register (GPIOF_PER)	16	R/W	See section	36.2.4/973
E254	GPIO Interrupt Assert Register (GPIOF_IAR)	16	R/W	0000h	36.2.5/974
E255	GPIO Interrupt Enable Register (GPIOF_IENR)	16	R/W	0000h	36.2.6/974
E256	GPIO Interrupt Polarity Register (GPIOF_IPOLR)	16	R/W	0000h	36.2.7/975
E257	GPIO Interrupt Pending Register (GPIOF_IPR)	16	R	0000h	36.2.8/975
E258	GPIO Interrupt Edge Sensitive Register (GPIOF_IESR)	16	R/W	0000h	36.2.9/976
E259	GPIO Push-Pull Mode Register (GPIOF_PPMODE)	16	R/W	See section	36.2.10/ 976
E25A	GPIO Raw Data Register (GPIOF_RAWDATA)	16	R	Undefined	36.2.11/ 977
E25B	GPIO Drive Strength Control Register (GPIOF_DRIVE)	16	R/W	See section	36.2.12/ 978
E25C	GPIO Pull Resistor Type Select (GPIOF_PUS)	16	R/W	See section	36.2.13/ 978
E25D	Slew Rate Control Register (GPIOF_SRE)	16	R/W	See section	36.2.14/ 979

36.2.1 GPIO Pull Resistor Enable Register (GPIOx_PUR)

This read/write register is for internal pull resistor enabling and disabling. If the pin is configured as an output, this register is not used. Unimplemented bits read as 0.

The pull resistor is intended only to drive an undriven input pin to a known state. It is characteristically a very weak pull.

Address: Base address + 0h offset

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read Write	PU															
Reset	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*

* Notes:
- The GPIOD_PUR register resets to 001Dh. The other GPIOn_PUR registers reset to 0000h.

GPIOx_PUR field descriptions

Field	Description
PU	Pull Resistor Enable Bits 0 Pull resistor is disabled 1 Pull resistor is enabled

36.2.2 GPIO Data Register (GPIOx_DR)

This register holds data that comes either from the pin or the data bus. In other words, the register is the data interface between the pin and the data bus.

Data written to this register appears on the pins if the pins are configured as GPIO output. Data read from this register is the same as the read state on the pins if those pins are configured as GPIO input.

When the device comes out of reset, GPIO pins are configured as inputs with internal pull disabled. As a result, the reset value of DR's bits is undefined. However, if you configure the GPIO as outputs (the DDR[DD] bit is 1), then the default value of the corresponding DR[D] bit is 0.

Address: Base address + 1h offset

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read Write	D															
Reset	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*

* Notes:
- x = Undefined at reset.

GPIOx_DR field descriptions

Field	Description
D	Data Bits

36.2.3 GPIO Data Direction Register (GPIOx_DDR)

This read/write register configures the state of the pin as either input or output when the pin is configured as GPIO (the corresponding bit in the GPIO peripheral enable register is set to 0). When the register is set to 0, the pin is an input. When the register is set to 1, the pin is an output.

Address: Base address + 2h offset

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read Write	DD															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

GPIOx_DDR field descriptions

Field	Description
DD	Data Direction Bits 0 Pin is an input 1 Pin is an output

36.2.4 GPIO Peripheral Enable Register (GPIOx_PER)

This read/write register determines the configuration of the GPIO pins.

When the Peripheral Enable bitfield value in this register is 1, the GPIO module is configured for peripheral mode. In this mode, a peripheral controls the GPIO pin, and the data transfer direction depends on the function of the peripheral.

When the Peripheral Enable bitfield value is 0, the pin is configured for GPIO mode. In this mode, the corresponding GPIO Data Direction register controls the data flow.

If write protection (via the SIM PROT register) is implemented on an individual chip, then this register value cannot be changed after the write protect signal has been asserted.

Address: Base address + 3h offset

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read Write	PE															
Reset	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*

* Notes:
- The GPIOD_PER register resets to 001Fh. The other GPIOn_PER registers reset to 0000h.

GPIOx_PER field descriptions

Field	Description
PE	Peripheral Enable Bits 0 Pin is for GPIO (GPIO mode) 1 Pin is for peripheral (peripheral mode)

36.2.5 GPIO Interrupt Assert Register (GPIOx_IAR)

This read/write register is only for software testing of a software interrupt capability. When the bit is set to 1, an interrupt is asserted. The interrupt is generated continually until this bit is cleared. Clear the bits in the register by writing 0s.

Address: Base address + 4h offset

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read Write	IA															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

GPIOx_IAR field descriptions

Field	Description
IA	Interrupt Assert Bits 0 Deassert software interrupt 1 Assert software interrupt

36.2.6 GPIO Interrupt Enable Register (GPIOx_IENR)

This read/write register enables or disables the edge interrupt from each GPIO pin. Set a bit to 1 to enable the interrupt for the associated GPIO pin. The interrupt is recorded in the corresponding GPIO Interrupt Pending register.

Address: Base address + 5h offset

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read Write	IEN															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

GPIOx_IENR field descriptions

Field	Description
IEN	Interrupt Enable Bits

GPIOx_IENR field descriptions (continued)

Field	Description
	0 External Interrupt is disabled 1 External Interrupt is enabled

36.2.7 GPIO Interrupt Polarity Register (GPIOx_IPOLR)

This read/write register is used for polarity detection caused by any external interrupts. The interrupt at the pin is active low when this register is set to 1 (falling edge causes the interrupt). The interrupt seen at the pin is active high when this register is set to 0 (rising edge causes the interrupt).

Address: Base address + 6h offset

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read Write	IPOL															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

GPIOx_IPOLR field descriptions

Field	Description
IPOL	Interrupt Polarity Bits 0 Interrupt occurred on rising edge 1 Interrupt occurred on falling edge

36.2.8 GPIO Interrupt Pending Register (GPIOx_IPR)

This read-only register is used to record any incoming interrupts. The user can read this register to determine which pin has caused the interrupt. This register can be cleared by writing 1s into the GPIO Interrupt Edge Sensitive register if the interrupt is caused by a pin, or by writing 0s into the GPIO Interrupt Assert register if the interrupt is caused by software.

Address: Base address + 7h offset

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	IP															
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

GPIOx_IPR field descriptions

Field	Description
IP	Interrupt Pending Bits 0 No Interrupt 1 Interrupt occurred

36.2.9 GPIO Interrupt Edge Sensitive Register (GPIOx_IESR)

When an edge is detected by the edge detector circuit and the GPIO Interrupt Enable register's field is set to 1, this register's field records the interrupt. This read/write register clears the corresponding Interrupt Pending bit by writing 1 to the appropriate Interrupt Edge Sensitive bit. Writing 0 to an Interrupt Edge Sensitive bit is ignored.

Address: Base address + 8h offset

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read Write	IES															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

GPIOx_IESR field descriptions

Field	Description
IES	Interrupt Edge-Sensitive Bits 0 No edge detected if read; no effect if writing 1 An edge detected if read; clear corresponding Interrupt Pending bit if writing

36.2.10 GPIO Push-Pull Mode Register (GPIOx_PPMODE)

This register can be used to explicitly set each output driver to either push-pull or open-drain mode. If write protection (via the SIM PROT register) is implemented on an individual device, then this register value cannot be changed after the write protect signal is asserted.

NOTE

Open-drain mode can be used to tri-state any pin on the GPIO port without switching that pin to input mode. This capability is useful for some applications, including a keypad interface.

Address: Base address + 9h offset

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read Write	PPMODE															
Reset	1*	1*	1*	1*	1*	1*	1*	1*	1*	1*	1*	1*	1*	1*	1*	1*

* Notes:
- The reset value of the register for each port is as follows:
  - GPIOA_PUS: 0FFFh
  - GPIOB_PUS: 0FFFh
  - GPIOC_PUS: FFFFh
  - GPIOD_PUS: 00FFh
  - GPIOE_PUS: 03FFh
  - GPIOF_PUS: FFFFh
  - GPIOG_PUS: 0FFFh

GPIOx_PPMODE field descriptions

Field	Description
PPMODE	Push-Pull Mode Bits 0 Open Drain Mode 1 Push-Pull Mode

36.2.11 GPIO Raw Data Register (GPIOx_RAWDATA)

This read-only register gives the core direct access to the logic values on each GPIO pin, even when pins are not in GPIO mode. Values are not clocked and are subject to change at any time. Read several times to ensure a stable value. The reset value of this register depends on the default PIN state.

Address: Base address + Ah offset

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	RAW_DATA															
Write																
Reset	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*

* Notes:
- x = Undefined at reset.

GPIOx_RAWDATA field descriptions

Field	Description
RAW_DATA	Raw Data Bits

36.2.12 GPIO Drive Strength Control Register (GPIOx_DRIVE)

This register can be used to explicitly set the drive strength of each output driver. If write protection (via the SIM PROT register) is implemented on an individual device, then this register value cannot be changed after the write protect signal is asserted.

Address: Base address + Bh offset

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read Write	DRIVE															
Reset	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*

* Notes:
- The GPIOD_DRIVE register resets to 000Ah. The other GPIOn_DRIVE registers reset to 0000h.

GPIOx_DRIVE field descriptions

Field	Description
DRIVE	Drive Strength Selector Bits 0 Low drive strength 1 High drive strength

36.2.13 GPIO Pull Resistor Type Select (GPIOx_PUS)

This register can be used to explicitly set the pull resistor type for each GPIO.

Address: Base address + Ch offset

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read Write	PUS															
Reset	1*	1*	1*	1*	1*	1*	1*	1*	1*	1*	1*	1*	1*	1*	1*	1*

* Notes:
- The reset value of the register for each port is as follows:
  - GPIOA_PUS: 0FFFh
  - GPIOB_PUS: 0FFFh
  - GPIOC_PUS: FFFFh
  - GPIOD_PUS: 00FBh
  - GPIOE_PUS: 03FFh
  - GPIOF_PUS: FFFFh
  - GPIOG_PUS: 0FFFh

GPIOx_PUS field descriptions

Field	Description
PUS	Pull Resistor Type Select Bits Note that the pull resistor type is ignored in open-drain mode (PPMODE==0) and, if the pull resistor is enabled, a pullup resistor is used. 0 Pulldown resistor 1 Pullup resistor

36.2.14 Slew Rate Control Register (GPIOx_SRE)

This register determines if output slew rate control is enabled for the associated GPIO port pin.

Address: Base address + Dh offset

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read Write	SRE															
Reset	1*	1*	1*	1*	1*	1*	1*	1*	1*	1*	1*	1*	1*	1*	1*	1*

* Notes:
- The reset value of the register for each port is as follows:
  - GPIOA_PUS: 0FFFh
  - GPIOB_PUS: 0FFFh
  - GPIOC_PUS: FFFFh
  - GPIOD_PUS: 00F5h
  - GPIOE_PUS: 03FFh
  - GPIOF_PUS: FFFFh
  - GPIOG_PUS: 0FFFh

GPIOx_SRE field descriptions

Field	Description
SRE	Slew Rate Enable 0 Slew rate is enabled (the turn-on time of the output transistor is faster) 1 Slew rate is disabled (the turn-on time of the output transistor is slower)

36.3 Functional Description

The following block diagram illustrates the logic associated with just one of the bits in each GPIO port. Each GPIO pin can be configured as:

- An input, with or without pull resistor functions
- An edge interrupt
- An output with push-pull or open-drain mode

The GPIO's pull resistor is enabled by writing to the PUR register. The resistor type is selected in the PUS register. When the pin is configured for a peripheral function, the pull resistors are controlled by the PUR register and the direction is specified by the peripheral used. If the pin is set to be an output, the pull resistor is disabled. In open-drain mode, the PUS register is ignored, and only the pullup resistor type is available.

A pin may have several peripheral functions, one of which may be an analog function. To access its analog function, the pin must be in peripheral mode with an analog input enabled. Selecting between an analog peripheral or a digital peripheral is controlled by the GPIO Peripheral Select (GPSn) registers in the System Integration Module (SIM). When the GPIO is in peripheral mode and its analog peripheral function is selected, the digital output buffer and pull resistor are disabled. The digital input buffer is also disconnected from the pin so that the digital input does not respond to analog voltages on the pin.

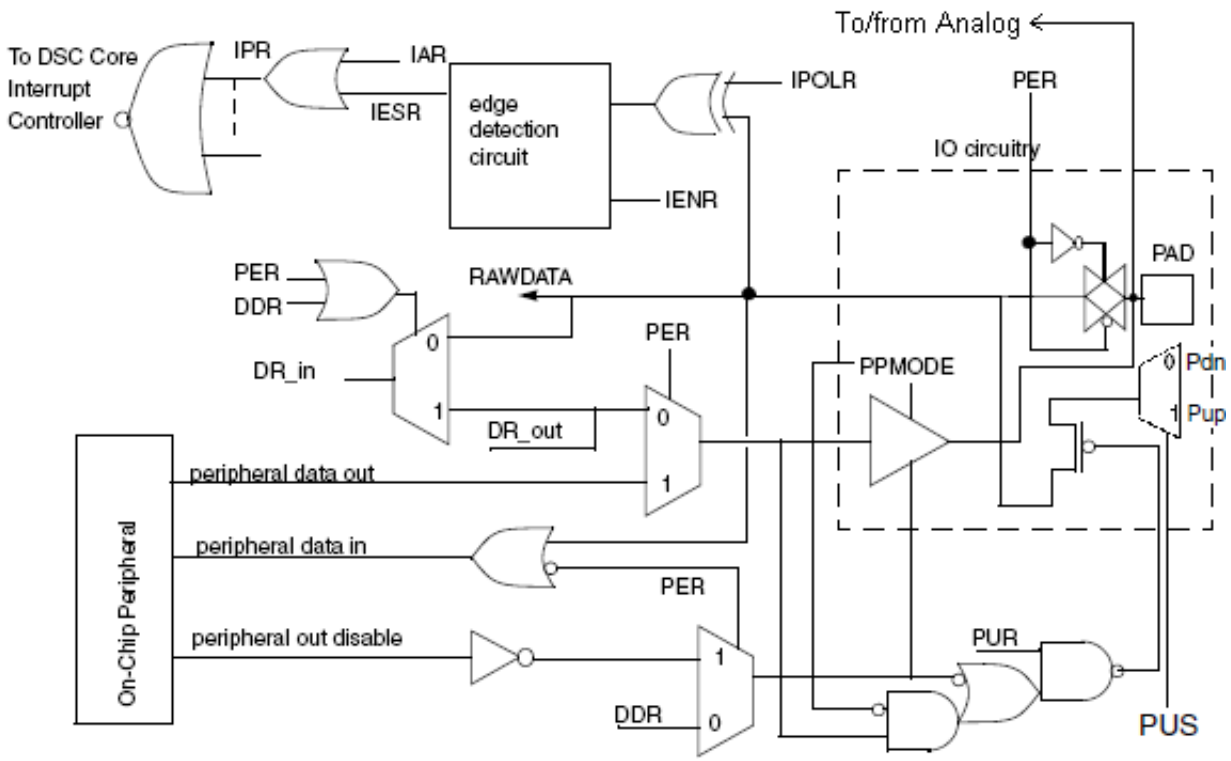


Figure 36-1. Bit View of the GPIO Logic with a Multiplex of Analog Input

36.4 Interrupts

The GPIO module has two types of interrupts:

1. Software interrupt

Write ones to the interrupt assert register to generate the software interrupt. The interrupt pending register records the value of the interrupt assert register. Clear the the interrupt pending register by writing zeroes to the the interrupt assert register.

NOTE

When a software interrupt is asserted, the interrupt polarity register, interrupt edge sensitive register, and the interrupt enable register must be zero to guarantee that the interrupt registered in the interrupt pending register is due only to the interrupt assert register.

2. Hardware interrupt from the pin

   When a GPIO pin is used as an external interrupt source, its IEN bit in the interrupt enable register is set to 1 and the interrupt assert register must be set to 0. The interrupt polarity register must be set to 1 for a falling edge interrupt and to 0 for a rising edge interrupt. When the signal at the pin transitions from high to low or low to high, the value is seen at the interrupt edge sensitive register and recorded by the interrupt pending register.

The interrupt signals in each port are ORed together, presenting only a single interrupt per port to the interrupt controller. The interrupt service routine must then check the contents of the interrupt pending register to determine which pin(s) caused the interrupt. External interrupt sources do not need to remain asserted because the detection mechanism is edge sensitive.

36.5 Clocks and Resets

The GPIO module runs at standard system bus speeds and assumes reset states as defined in the device data sheet. Reset occurs whenever any source of system reset occurs (POR, external reset, COP reset, and so on).

Appendix A
Revision History

A.1 Revision History

Table A-1. Revision history

Revision/ Date	Chapter	Changes
Rev 2, Sept 2016		Added support for MWCT1013A devices (was previously MWCT1003A devices only), changing MWCT1003 to MWCT10x3A, to indicate support for MWCT1013A devices.
	Chapter 2, Introduction	Almost all changes are here.
	Appendix Revision History	Added.

How to Reach Us:

Home Page:
nxp.com

Web Support:
nxp.com/support

Information in this document is provided solely to enable system and software implementers to use NXP products. There are no express or implied copyright licenses granted hereunder to design or fabricate any integrated circuits based on the information in this document. NXP reserves the right to make changes without further notice to any products herein.

NXP makes no warranty, representation, or guarantee regarding the suitability of its products for any particular purpose, nor does NXP 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 NXP 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. NXP does not convey any license under its patent rights nor the rights of others. NXP sells products pursuant to standard terms and conditions of sale, which can be found at the following address: nxp.com/SalesTermsandConditions.

NXP, the NXP logo, NXP SECURE CONNECTIONS FOR A SMARTER WORLD, COOLFLUX, EMBRACE, GREENCHIP, HITAG, I2C BUS, ICODE, JCOP, LIFE VIBES, MIFARE, MIFARE CLASSIC, MIFARE DESFire, MIFARE PLUS, MIFARE FLEX, MANTIS, MIFARE ULTRALIGHT, MIFARE4MOBILE, MIGLO, NTAG, ROADLINK, SMARTLX, SMARTMX, STARPLUG, TOPFET, TRENCHMOS, UCODE, Freescale, the Freescale logo, AltiVec, C-5, CodeTest, CodeWarrior, ColdFire, ColdFire+, C-Ware, the Energy Efficient Solutions logo, Kinetis, Layerscape, MagniV, mobileGT, PEG, PowerQUICC, Processor Expert, QorIQ, QorIQ Qonverge, Ready Play, SafeAssure, the SafeAssure logo, StarCore, Symphony, VortiQa, Vybrid, Airfast, BeeKit, BeeStack, CoreNet, Flexis, MXC, Platform in a Package, QUICC Engine, SMARTMOS, Tower, TurboLink, and UMEMS are trademarks of NXP B.V. All other product or service names are the property of their respective owners. ARM, AMBA, ARM Powered, Artisan, Cortex, Jazelle, Keil, SecurCore, Thumb, TrustZone, and µVision are registered trademarks of ARM Limited (or its subsidiaries) in the EU and/or elsewhere. ARM7, ARM9, ARM11, big.LITTLE, CoreLink, CoreSight, DesignStart, Mali, mbed, NEON, POP, Sensinode, Socrates, ULINK and Versatile are trademarks of ARM Limited (or its subsidiaries) in the EU and/or elsewhere. All rights reserved. Oracle and Java are registered trademarks of Oracle and/or its affiliates. The Power Architecture and Power.org word marks and the Power and Power.org logos and related marks are trademarks and service marks licensed by Power.org.

© 2014–2016 NXP B.V.

Document Number MWCT10x3ARM
Revision 2, 09/2016